Bases De Données

. Conception d'une BDD Relationnelle
Modèle Conceptuel de Données
Passage au Modèle Logique de Données

. Construction d'une BDD
Langage de définition SQL : create, drop, alter
SQBD : SQL Anywhere

. Manipulation d'une BDD
Modèle relationnelle
Langage de manipulation SQL : insert, update, delete, select
Langage de définition SQL : create view

. Traitements dans les BDD
Triggers
Procédures stockées

. Sécurités des BDD
Journalisation, transactions : commit, rollback
Droits d'accès : grant, revoke

. Accès aux BDD dans les langages hôtes
Visual Basic – SQL Server

1 - Conception d'une BDD Relationnelle
Modèle Conceptuel de Données

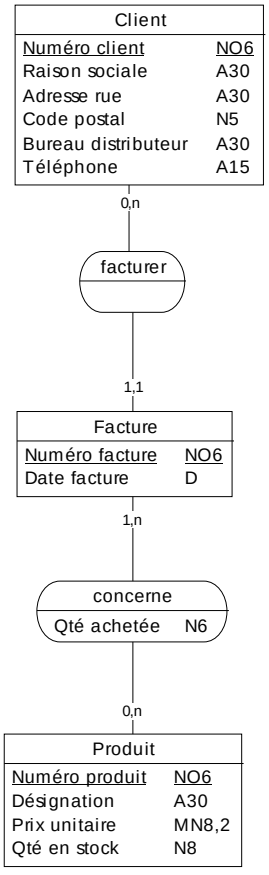
OBJET : reflet d'une ENTITE statique manipulée par l'organisme, dotée d'une existence propre, dont chaque occurrence est identifiable par une donnée particulière.

RELATION : ENTITE dont l'existence des occurrences dépend de l'existence des occurrences d'objets qui la composent.

PROPRIETE : plus petit ELEMENT logique d'information, qui a un sens en lui-même et dont la valeur caractérise partiellement une occurrence d'OBJET ou de RELATION.

Remarque : tout objet doit comporter une propriété à chaque valeur de laquelle ne correspondra qu'une occurrence de l'OBJET : c'est l'**IDENTIFIANT**.

CARDINALITE : nombre mini et maxi d'occurrences d'une RELATION pour une occurrence d'OBJET ; seuls 3 chiffres sont significatifs : 0, 1 et n.



OBJET CLIENT

RELATION FACTURER

OBJET FACTURE

RELATION CONCERNER

OBJET PRODUIT

1 - Conception d'une BDD Relationnelle
Cas Note

Soit la liste des données recensées dans un établissement scolaire et présentées par ordre alphabétique :

- Adresse de l'élève
- Matière enseignée
- Nombre d'heures
- Nom de la classe
- Nom de l'élève
- Nom du professeur
- Note
- Numéro de salle
- Prénom de l'élève

Règles de gestion :

A chaque classe est attribuée une et une salle de cours

Chaque matière n'est enseignée que par un et un seul professeur

Pour chaque classe et chaque matière est défini un nombre fixe d'heures de cours

A chaque élève est attribuée une seule note par matière

L'établissement gère les emplois du temps des professeurs et des élèves ainsi que le contrôle de connaissances.

1 - Conception d'une BDD Relationnelle
Cas Niche

Liste des propriétés :

- Adresse du propriétaire
- Date d'achat
- Date de concours
- Date de décès du chien
- Date de naissance du chien
- Nom du chien
- Nom du propriétaire
- Numéro de matricule du chien
- Race
- Résultat du concours
- Sexe du chien
- Type de concours
- Ville de concours

Il s'agit de construire un système permettant de connaître le cursus honorum des chiens de race participant à des concours, ainsi que leur généalogie.

1 - Conception d'une BDD Relationnelle
Cas FAP

L'organisme FAP pour un type d'activité d'entreprise donné est habilité à collecter le 1% de la formation professionnelle, cet organisme réalise lui-même de la formation. Les entreprises versent 1% des salaires bruts à FAP pour l'année écoulée avant le 1er mars en un ou plusieurs versements, ce qui leur octroie un crédit de formation du même montant.

FAP organise des stages de formation dont la durée varie de 1 à 10 jours ; chaque stage a des dates de stage, un nombre d'heures global, un coût, ...

Les entreprises y inscrivent des stagiaires (membres de l'entreprise), le coût correspondant du stage est débité du compte de l'entreprise, si le crédit de l'entreprise n'est pas suffisant une facture de formation (avec TVA) est adressée à l'entreprise à l'issue du stage (facture complète ou partielle).

Les stagiaires n'étant pas toujours présents durant toute la durée du stage, il est nécessaire à partir des feuilles de présence signées par les stagiaires de saisir les dates d'absences et le nombre d'absences correspondant.

En fin de stage, on édite une attestation de présence par entreprise ayant eu des stagiaires dans ce stage, donnant le nom du stage, les dates, le nombre d'heures du stage, et par stagiaire le nombre d'heures de présence, les dates d'absences avec les nombres d'heures d'absences correspondant à chaque date d'absence.

Cette édition est faite tous les mois pour les stages se terminant durant le mois et dont les absences furent enregistrées.

Tous les mouvements comptables doivent être conservés pour toutes les entreprises :
- versement du 1%
- enregistrement des débits
- règlements complémentaires

Suppression de l'historique une fois par an avec édition des positions de compte (reflet pour chaque entreprise des différentes écritures comptables).

En cours d 'année on peut vouloir éditer une position de compte pour une entreprise, une plage de numéros d'entreprise ou toutes les entreprises. La suppression des écritures ne se fait pas dans ce type d'édition.

1 - Conception d'une BDD Relationnelle
Cas Réservation Hôtel

Champ de l'étude : réservation des chambres d'hôtel des stations de sport d'hiver connues par un organisme de voyages.

Liste des propriétés :

- Adresse du client
- Capacité de l'hôtel
- Capacité de la chambre
- Catégorie de l'hôtel
- Date début de séjour
- Date fin de séjour
- Date émission de la réservation
- Degré de confort
- Distance gare SNCF – Station de ski
- Exposition de la chambre
- Gare SNCF
- Montant des arrhes versées
- Nom du client
- Nom de l'hôtel
- Numéro de client
- Numéro de la chambre
- Numéro de la réservation
- Numéro de téléphone du client
- Période tarifaire
- Prix journalier de la chambre
- Station de ski
- Type de chambre

Règles de gestion :

- Le prix est déterminé dans chaque station par la catégorie de l'hôtel, la période tarifaire et le type de chambre.

- Le type de chambre dépend de la capacité, du degré de confort et de l'exposition de la chambre.

1 - Conception d'une BDD Relationnelle
Passage au Modèle Logique

Contexte Relationnel :

ATTRIBUT : correspond à la définition de donnée ou de propriété

DOMAINE : ensemble des valeurs que peut prendre un attribut

TABLE : sous-ensemble du produit cartésien d'une liste de domaines ; toute occurrence, appelée TUPLE, est une suite évaluée des ATTRIBUTS qui composent la TABLE

Règles de passage :

. Tout OBJET devient une TABLE ; s'il ne porte que son identifiant, il peut ne pas donner lieu à transformation. La propriété identifiante devient alors la clé primaire de la table.

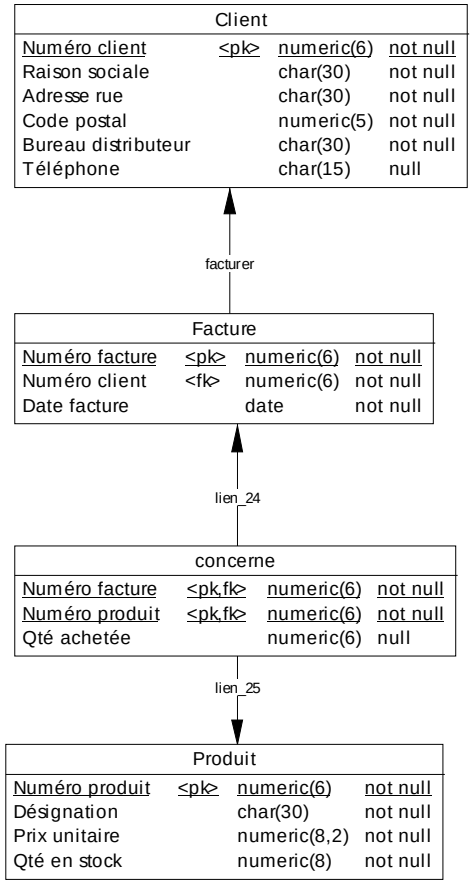
. Une RELATION BINAIRE de cardinalités x,1 – x,n (x valant 0 ou 1) provoque l'ajout dans la TABLE issue de l'OBJET de cardinalité x,1, d'un ATTRIBUT supplémentaire correspondant à l'identifiant de l'OBJET de cardinalité x,n nommé clé étrangère ; si la relation est porteuse de données, celles-ci passent dans cette même TABLE.

. Dans le cas d'une relation d'appartenance, la clé étrangère s'intégrera à la clé primaire.

. Une RELATION BINAIRE de cardinalités x,n – x,n donne lieu à la création d'une TABLE dont 2 des ATTRIBUTS correspondront aux identifiants des 2 objets.

. Par extension, une RELATION n-aire donnera une TABLE dont n des ATTRIBUTS correspondront aux identifiants des n OBJETS.

Modèle Logique de Données :



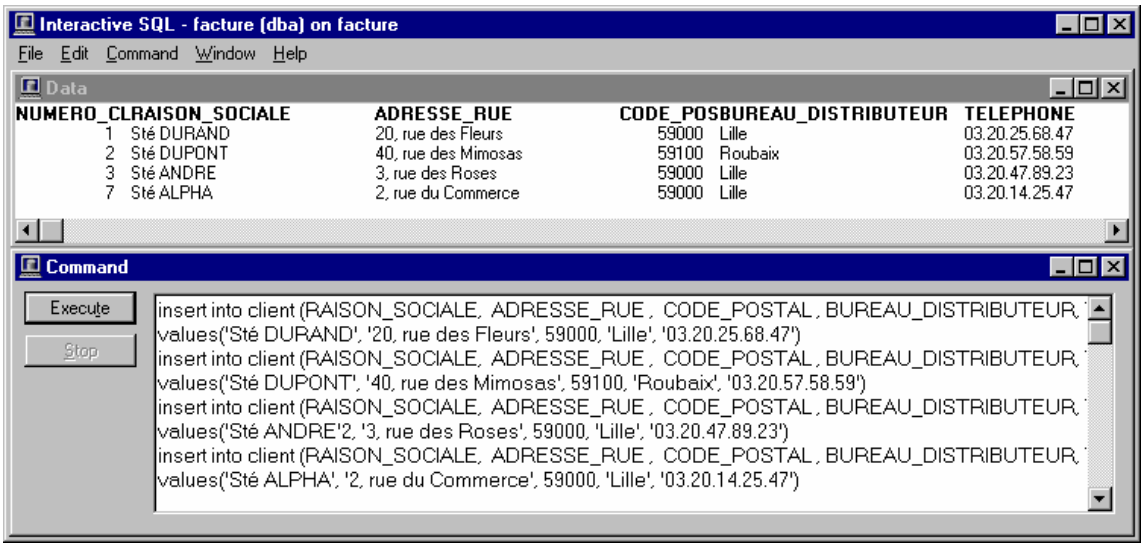
2 - Construction d'une BDD
Langage de définition SQL

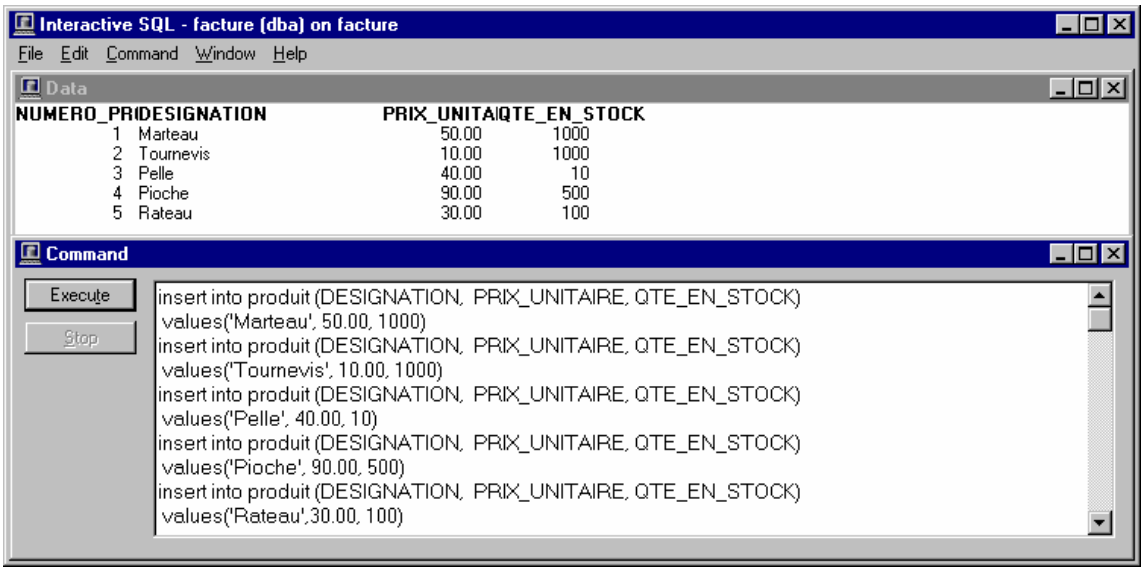
DBTOOL CREATE DATABASE
DBTOOL DROP DATABASE
CREATE TABLE
DROP TABLE
ALTER TABLE

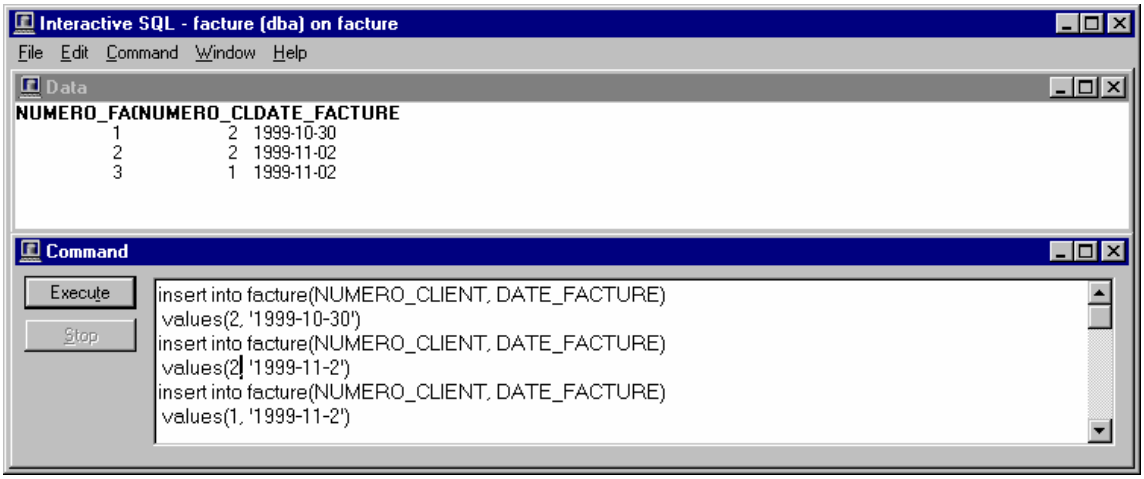
```
%% ============================================================
%%   Nom de la base   :  FACTURE
%%   Nom de SGBD      :  Sybase SQL Anywhere
%%   Date de création :  02/11/99  11:55
%% ============================================================

%% ============================================================
%%   Nom de la base   :  FACTURE
%% ============================================================
dbtool create database 'facture.db' transaction log to 'facture.log';
connect to 'facture.db' user "dba" identified by sql;

%% ============================================================
%%   Table : CLIENT
%% ============================================================
create table CLIENT
(
   NUMERO_CLIENT       numeric(6)          not null
      default AUTOINCREMENT,
   RAISON_SOCIALE      char(30)            not null,
   ADRESSE_RUE         char(30)            not null,
   CODE_POSTAL         numeric(5)          not null,
   BUREAU_DISTRIBUTEUR  char(30)           not null,
   TELEPHONE           char(15)                ,
   primary key (NUMERO_CLIENT)
);

%% ============================================================
%%   Table : PRODUIT
%% ============================================================
create table PRODUIT
(
   NUMERO_PRODUIT      numeric(6)          not null
      default AUTOINCREMENT,
   DESIGNATION         char(30)            not null,
   PRIX_UNITAIRE       numeric(8,2)        not null,
   QTE_EN_STOCK        numeric(8)          not null,
   primary key (NUMERO_PRODUIT)
```

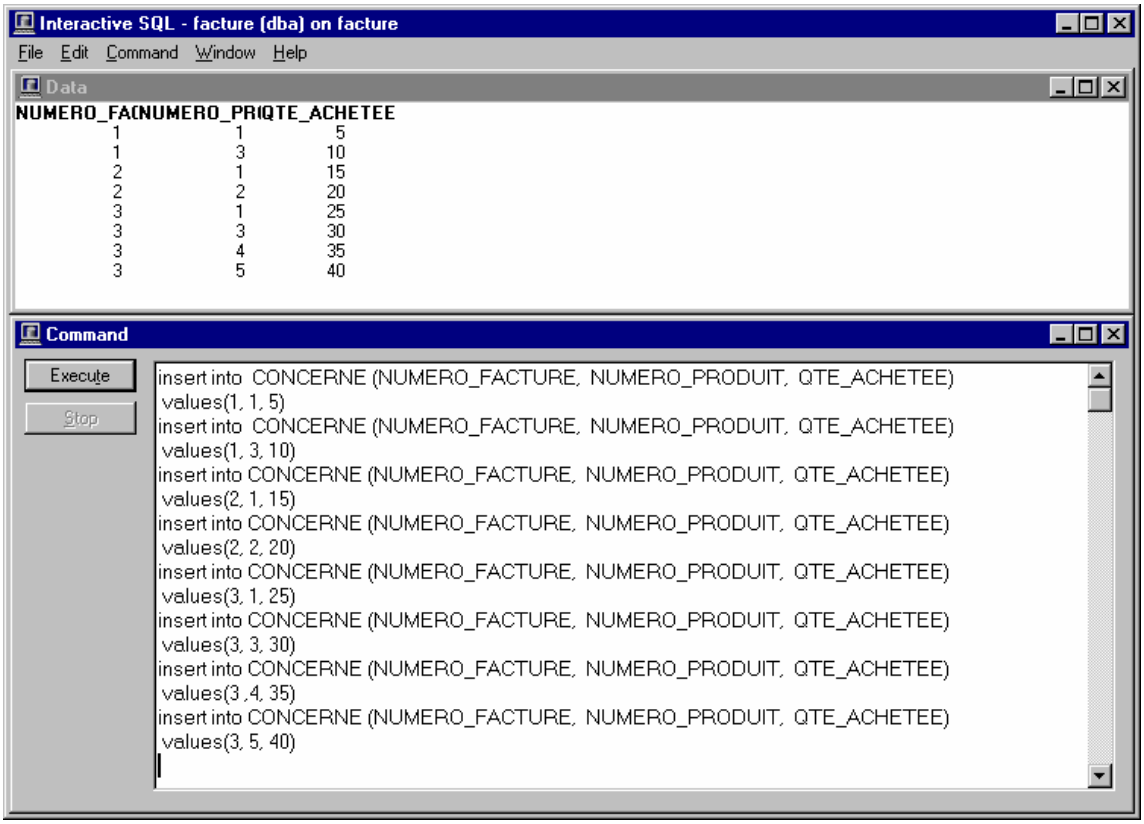
```
);

%% ==============================================================
%%   Table : FACTURE
%% ==============================================================
create table FACTURE
(
   NUMERO_FACTURE       numeric(6)           not null
      default AUTOINCREMENT,
   NUMERO_CLIENT        numeric(6)           not null,
   DATE_FACTURE         date                 not null,
   primary key (NUMERO_FACTURE)
);

%% ==============================================================
%%   Table : CONCERNE
%% ==============================================================
create table CONCERNE
(
   NUMERO_FACTURE       numeric(6)           not null,
   NUMERO_PRODUIT       numeric(6)           not null,
   QTE_ACHETEE          numeric(6)                ,
   primary key (NUMERO_FACTURE, NUMERO_PRODUIT)
);

alter table FACTURE
   add foreign key FK_FACTURE_ASSOC_3_CLIENT (NUMERO_CLIENT)
      references CLIENT (NUMERO_CLIENT) on update restrict on delete cascade;

alter table CONCERNE
   add foreign key FK_CONCERNE_LIEN_24_FACTURE (NUMERO_FACTURE)
      references FACTURE (NUMERO_FACTURE) on update restrict on delete cascade;

alter table CONCERNE
   add foreign key FK_CONCERNE_LIEN_25_PRODUIT (NUMERO_PRODUIT)
      references PRODUIT (NUMERO_PRODUIT) on update restrict on delete cascade;
```

3 - Manipulation d'une BDD
Langage de manipulation SQL
INSERT





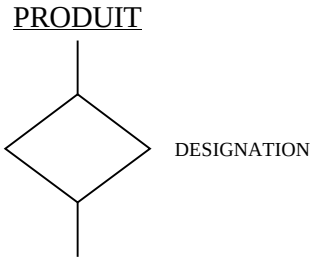


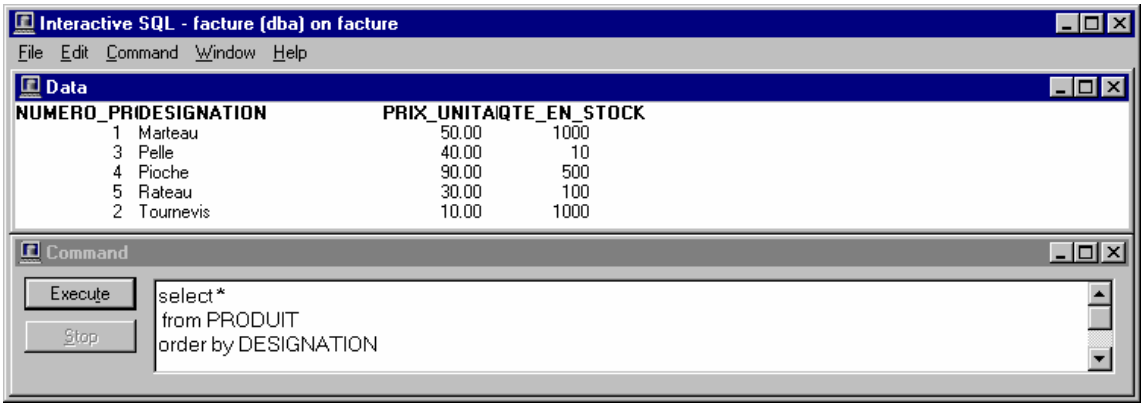


3 - Manipulation d'une BDD
Langage de manipulation SQL
SELECT

TRI

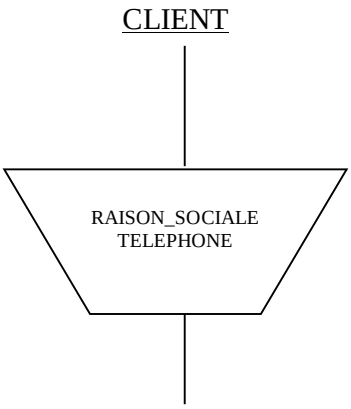
Opération consistant à trier les lignes

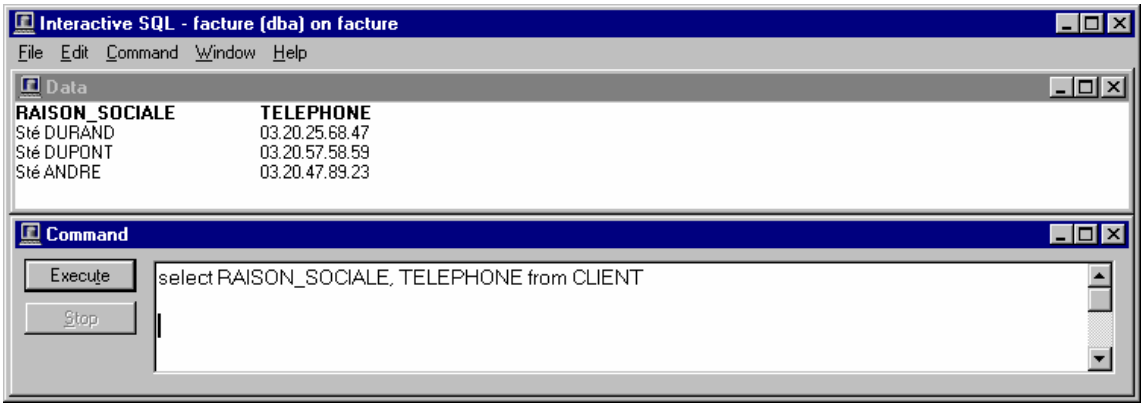




PROJECTION

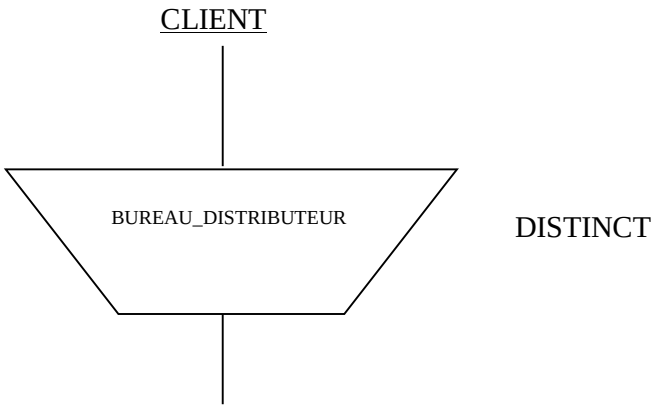
Opération consistant à éliminer les colonnes non utilisées

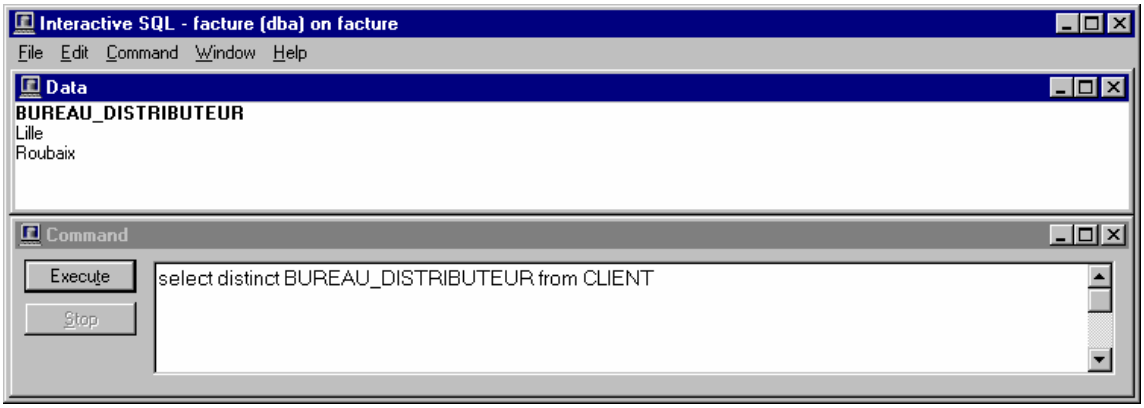




PROJECTION

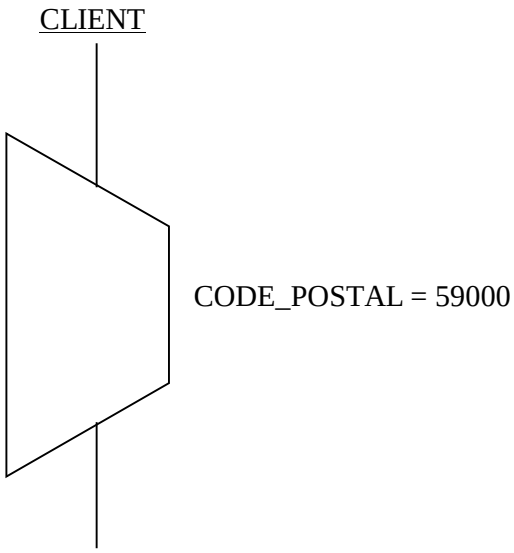
Opération consistant à éliminer les colonnes non utilisées et à éliminer les lignes en double

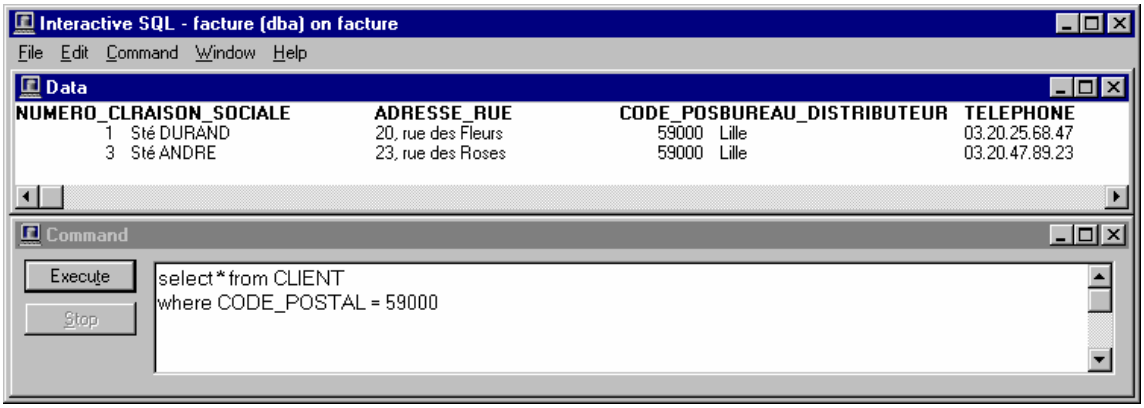




SELECTION

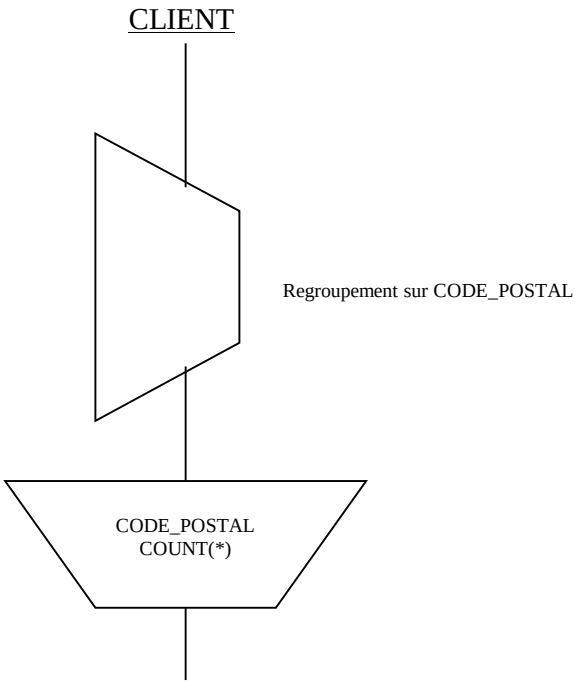
Opération consistant à sélectionner les lignes en fonction d'une condition

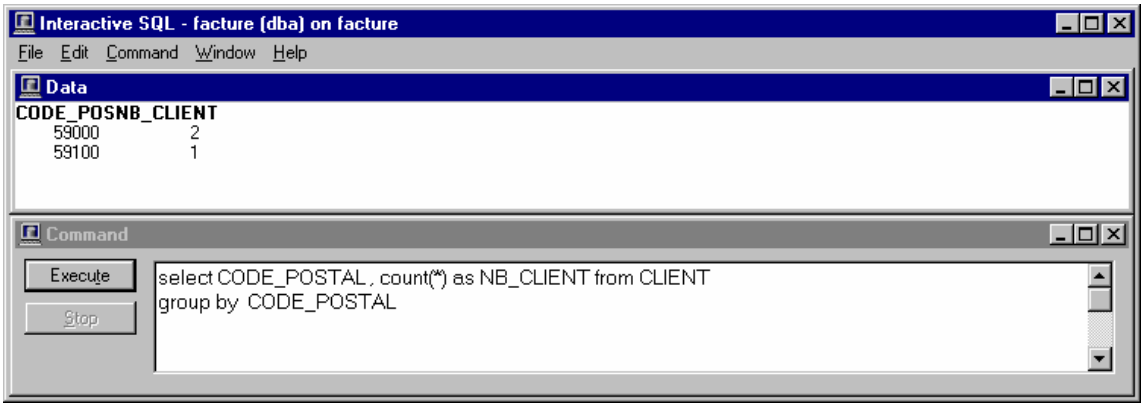




REGROUPEMENT

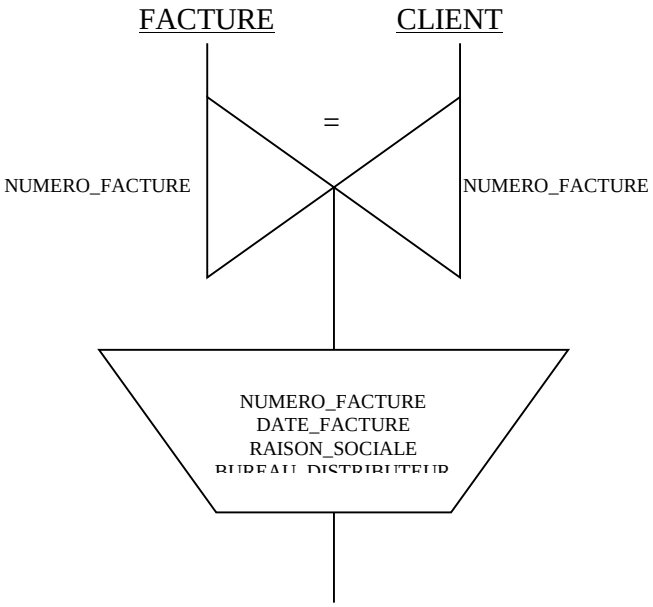
Opération consistant à regrouper les lignes et a effectuer des calculs liés au regroupement tels que moyenne, comptage et cumul.

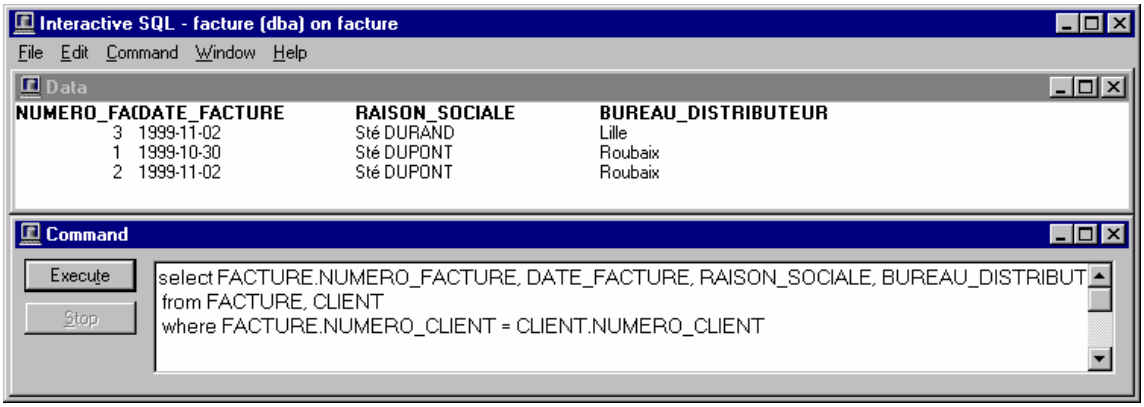




EQUI-JOINTURE

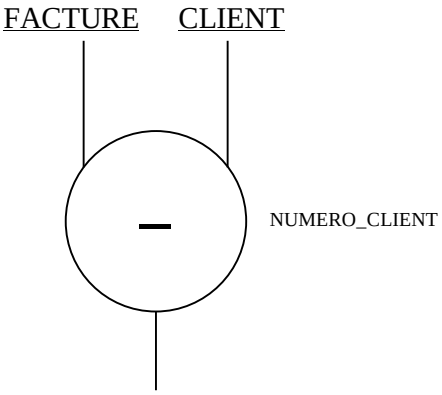
Opération consistant à rapprocher les lignes de 2 tables en fonction d'une condition d'égalité

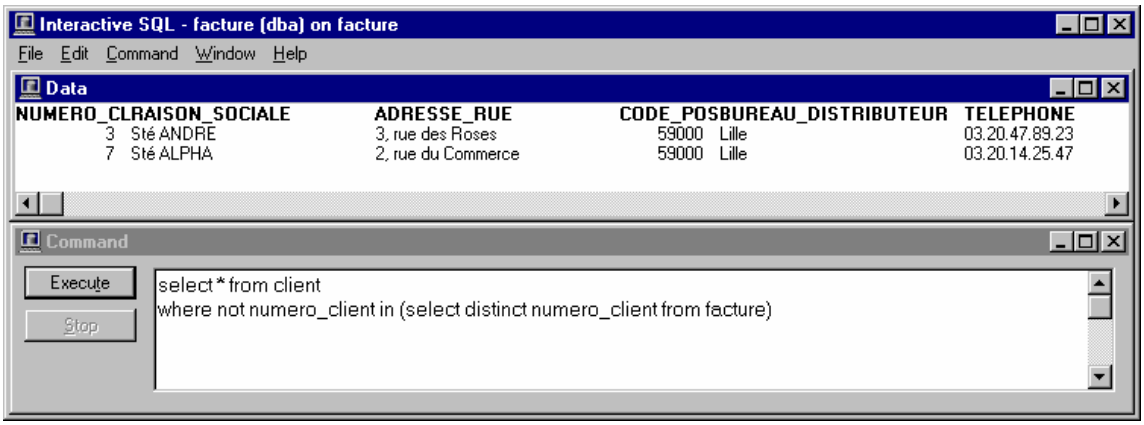




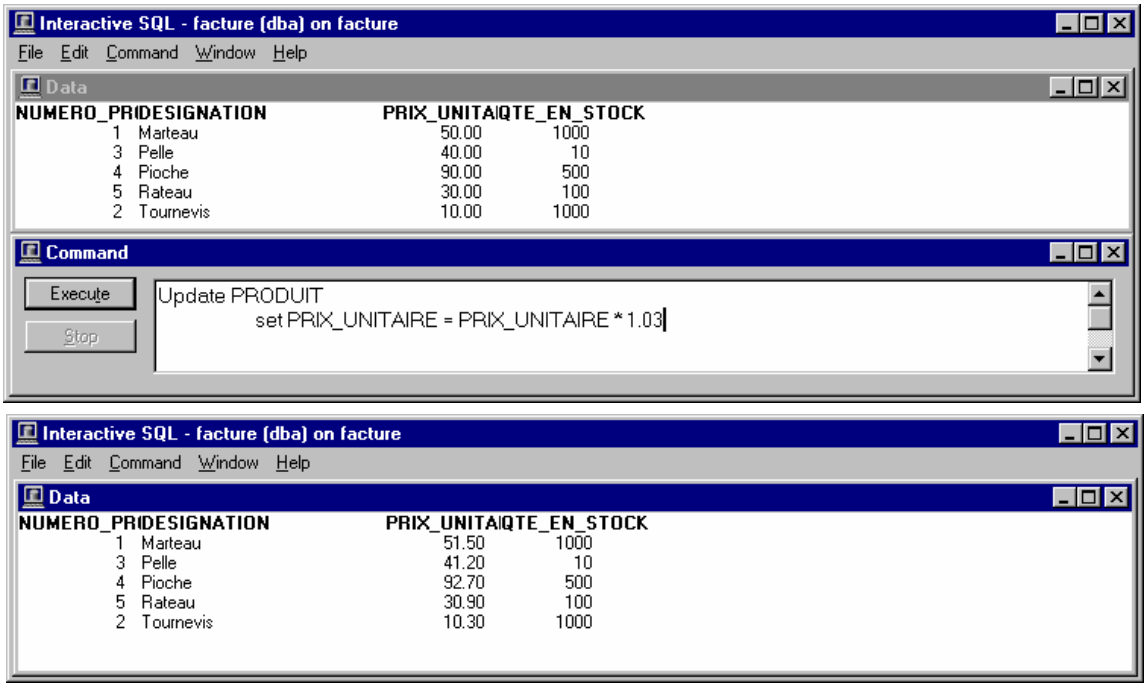
DIFFERENCE

Opération consistant à ne conserver que les lignes d'une table ne figurant pas dans une autre table

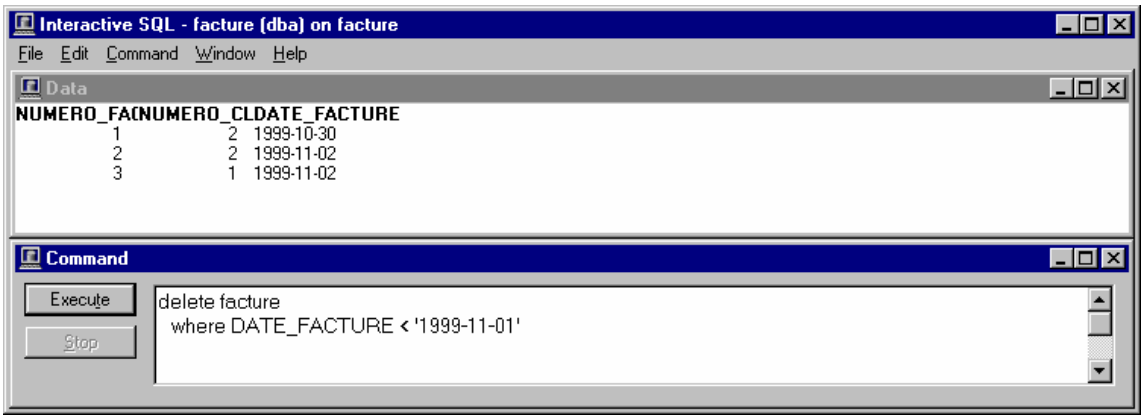


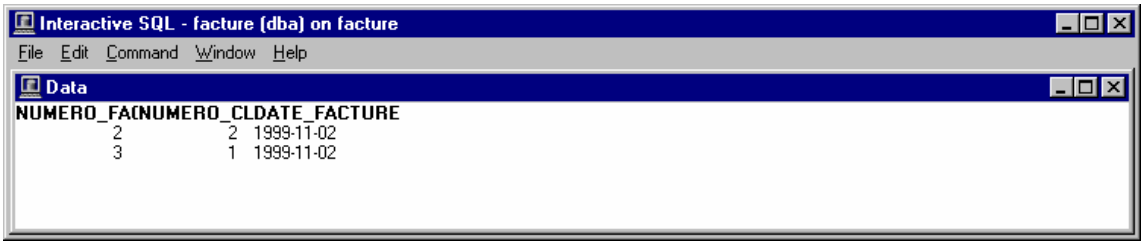


3 - Manipulation d'une BDD
Langage de manipulation SQL
UPDATE

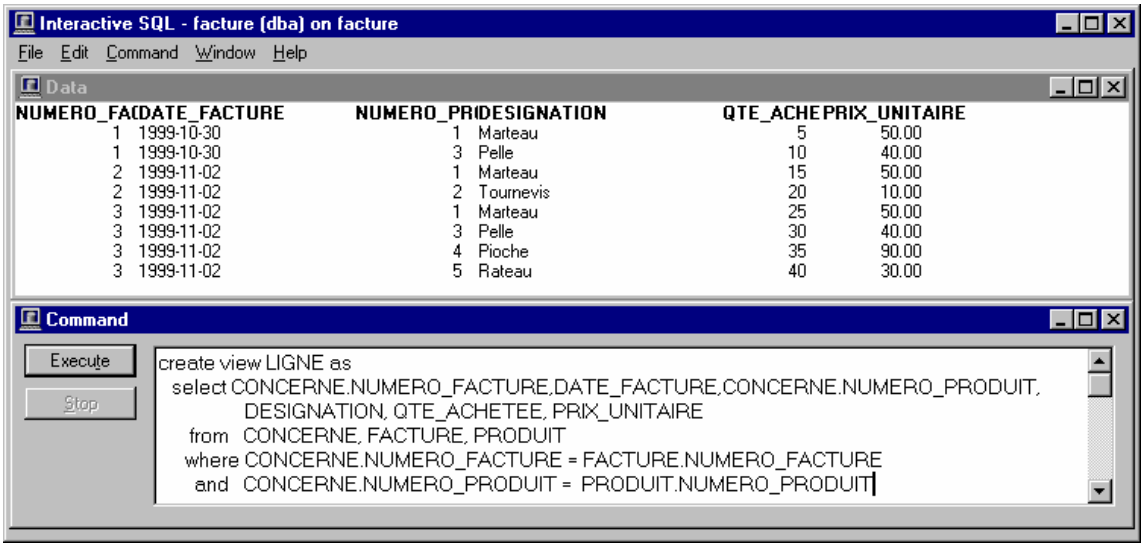


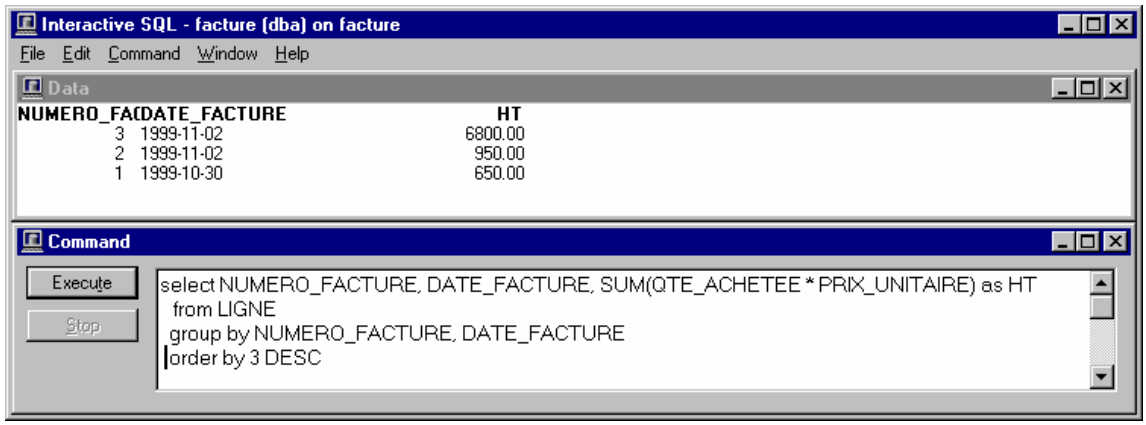
3 - Manipulation d'une BDD
Langage de manipulation SQL
DELETE





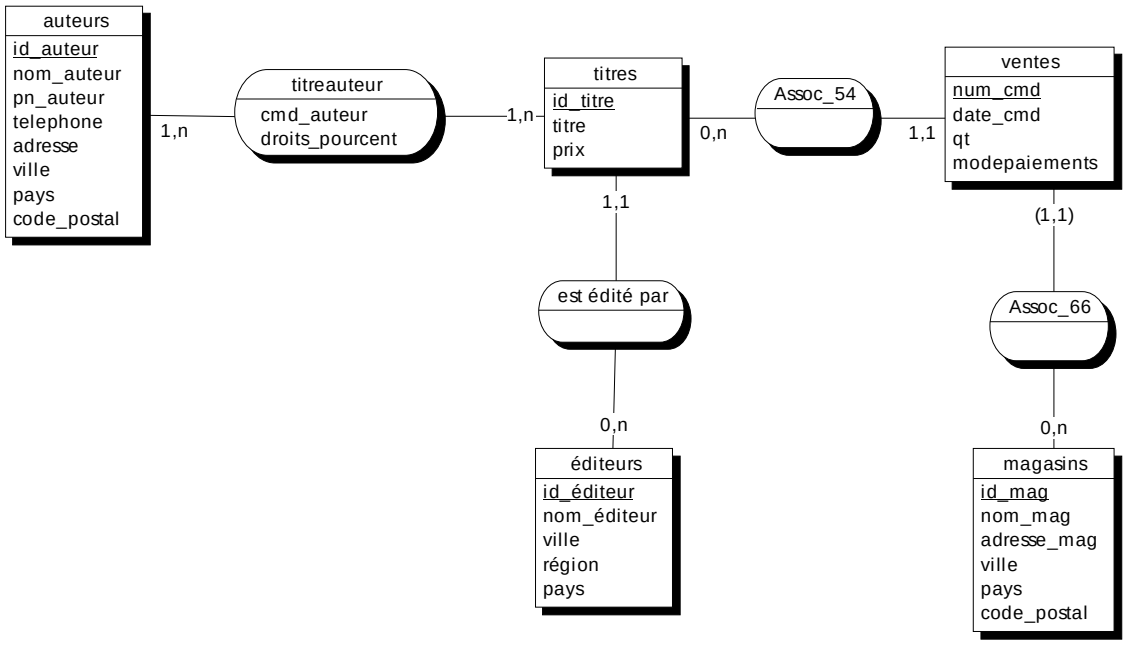
3 - Manipulation d'une BDD
Langage de définition SQL
CREATE VIEW



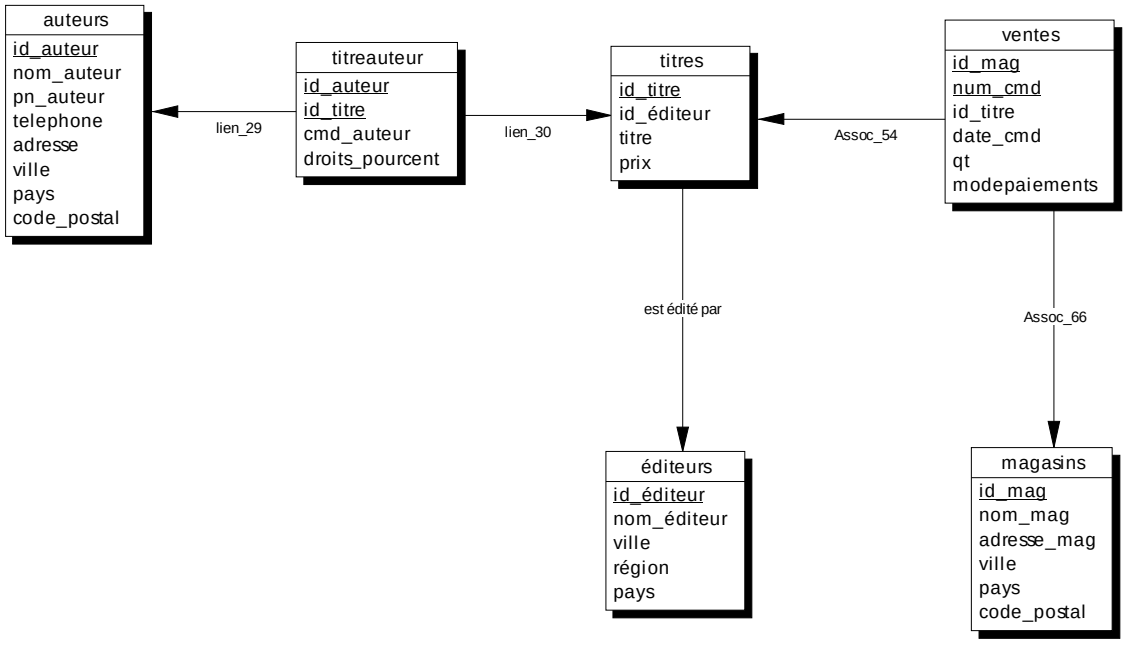


3 - Manipulation d'une BDD
Langage de définition SQL
Base De Données Pubs

Modèle Conceptuel de Données:



Modèle Logique de Données :



3 - Manipulation d'une BDD
Langage de définition SQL
Base De Données Pubs

Exercices :

. Liste des magasins français ('FR') dans l'ordre alphabétique des villes

. Liste des titres des auteurs étrangers

. Nombre de titres par auteur étranger

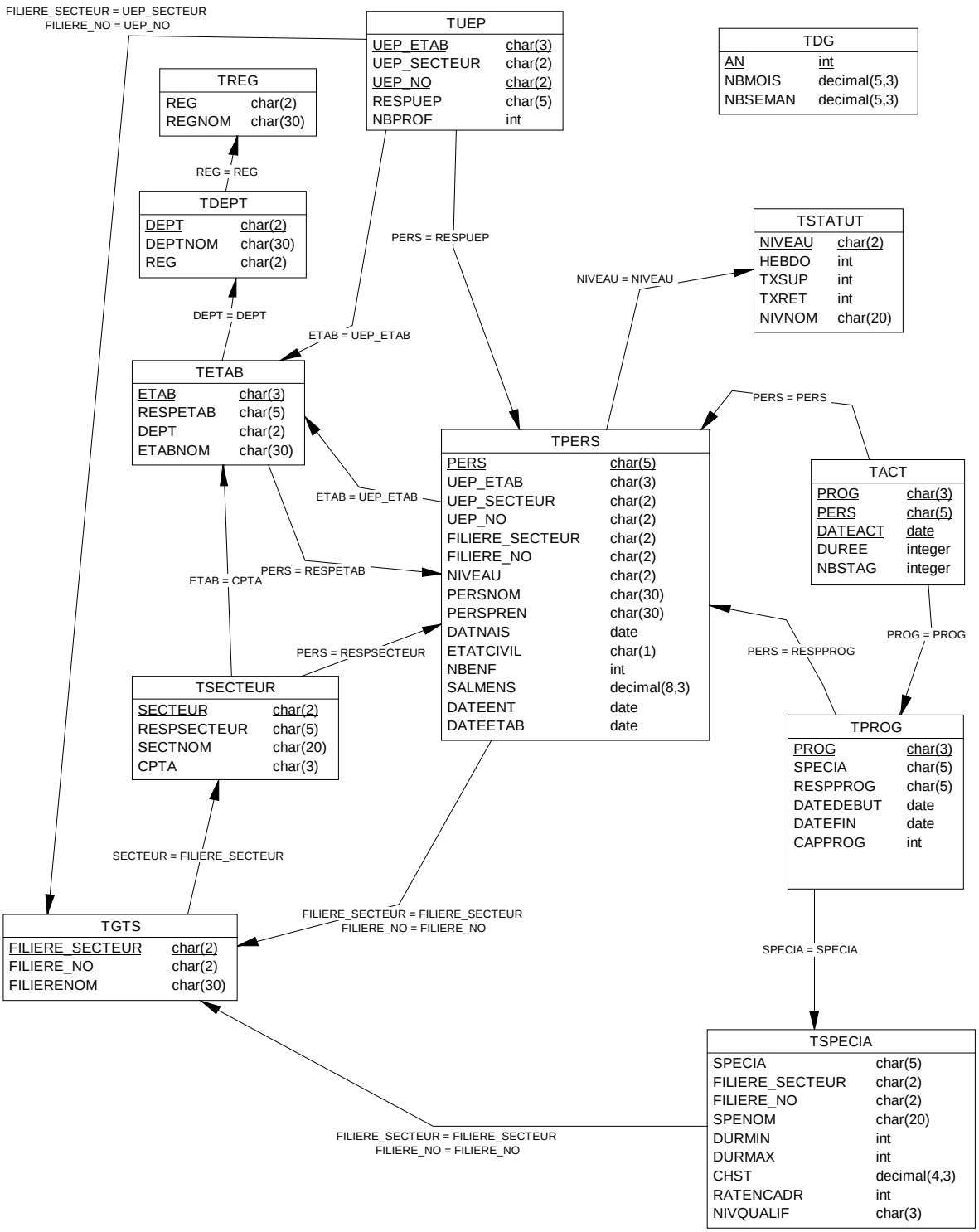
. Dans quel pays sont vendus les titres des auteurs étrangers (liste par auteur et par pays)

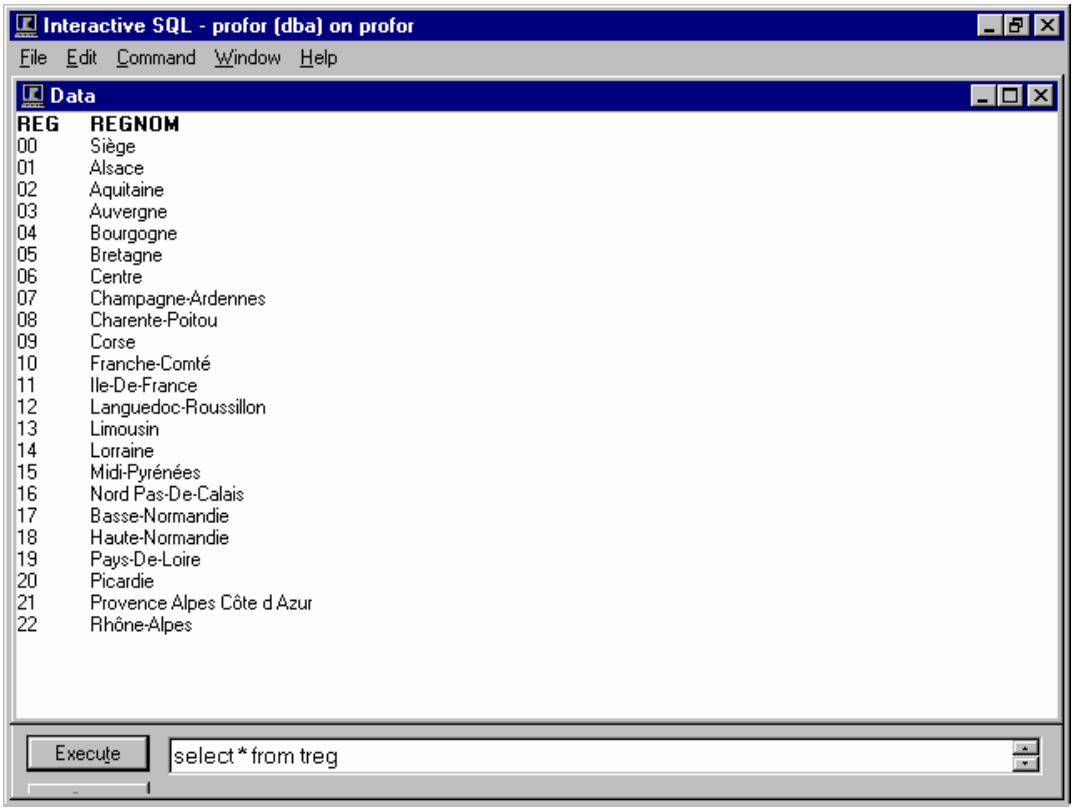
. Hit parade des ventes par auteur (liste décroissante, les ventes sont appréciées en quantité)

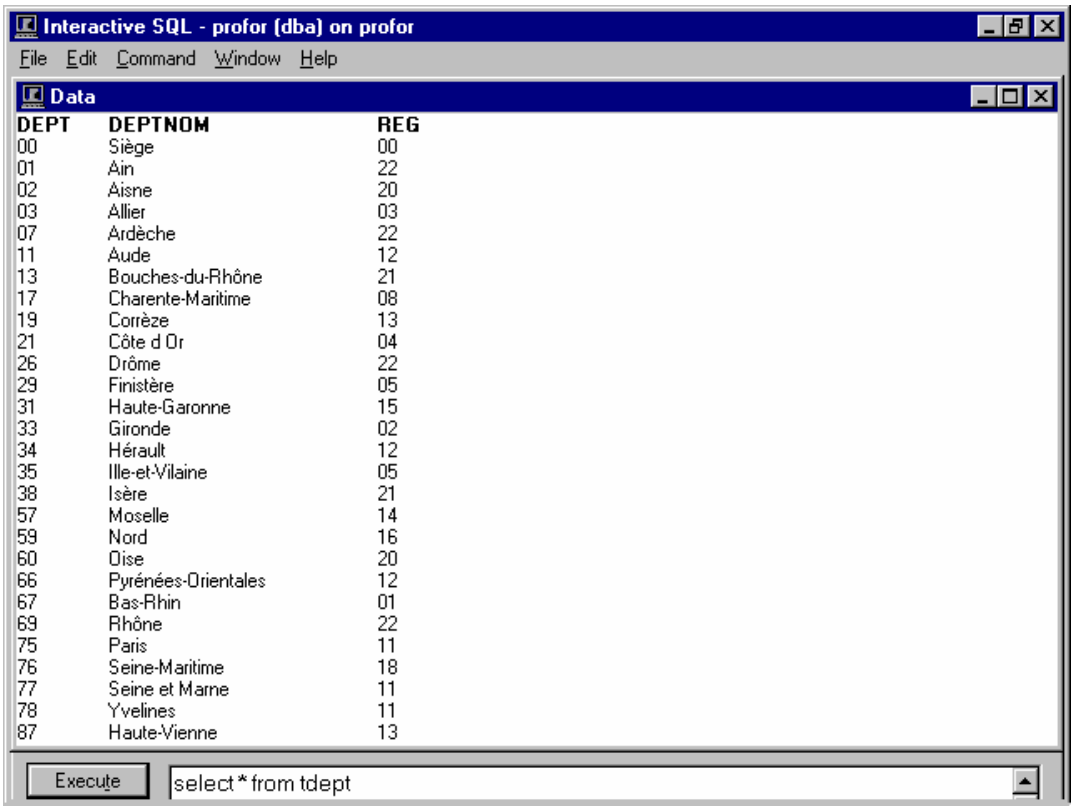
. Chez quels éditeurs sont publiés les auteurs (liste par auteur et éditeur)

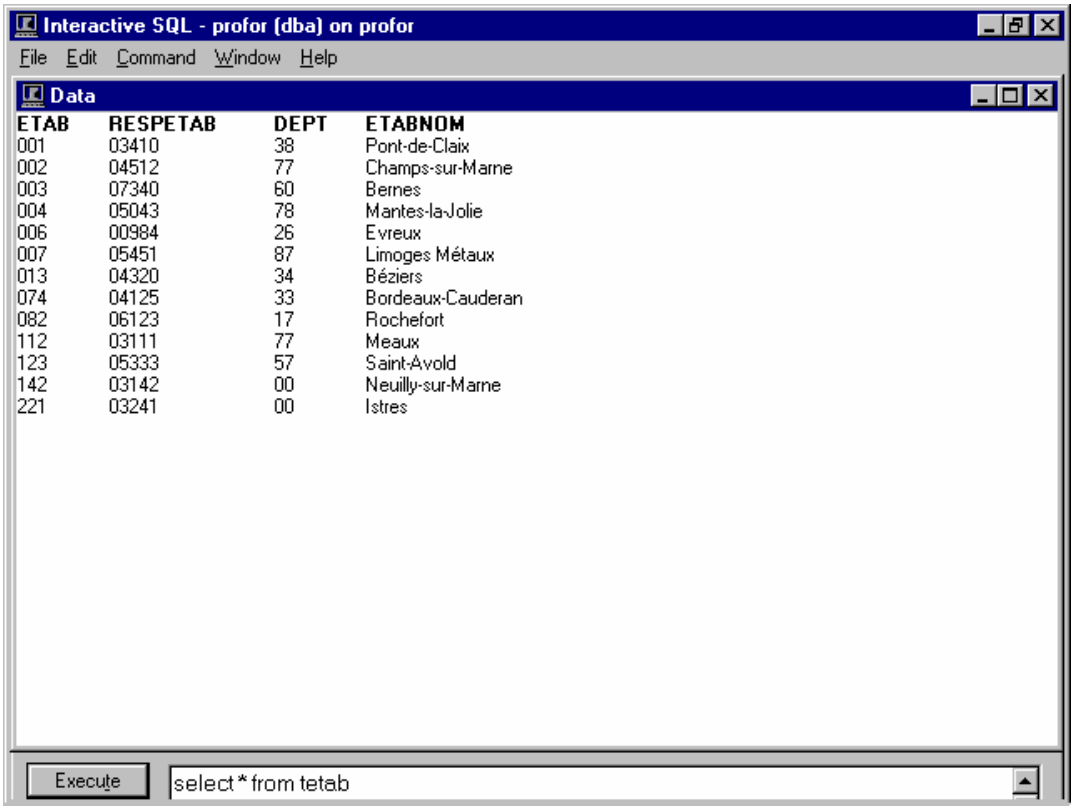
. Cumul des ventes par magasin et par mois

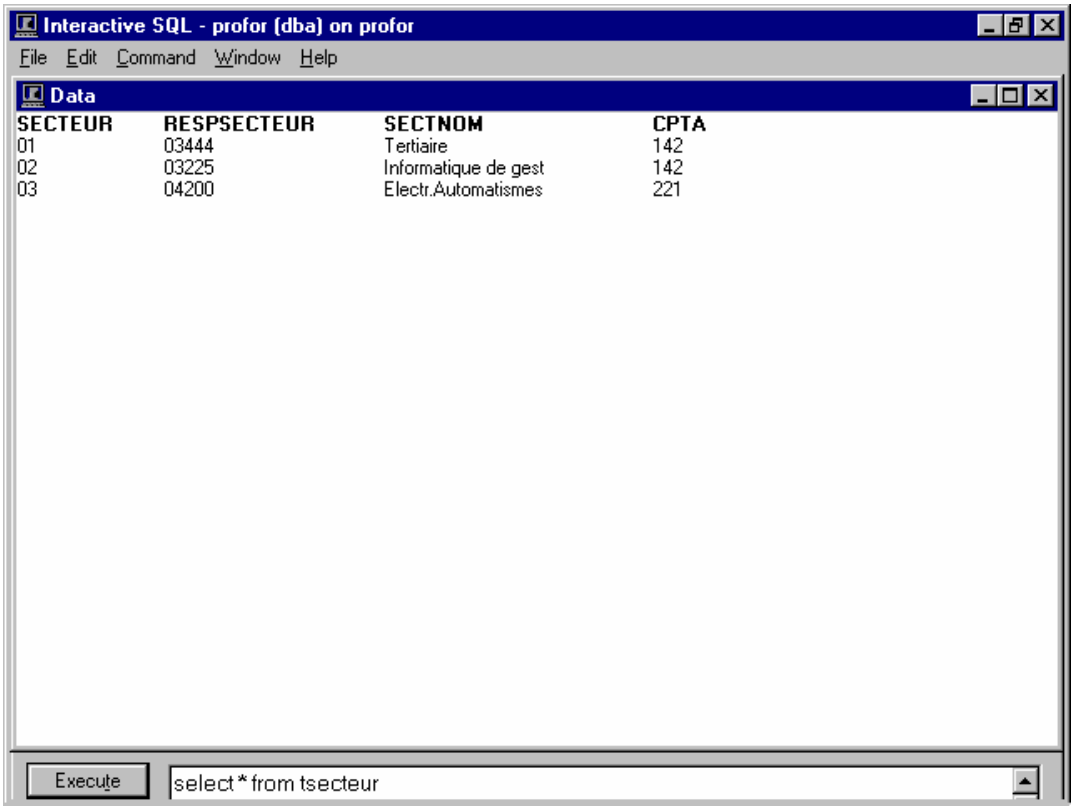
Base de données PROFOR

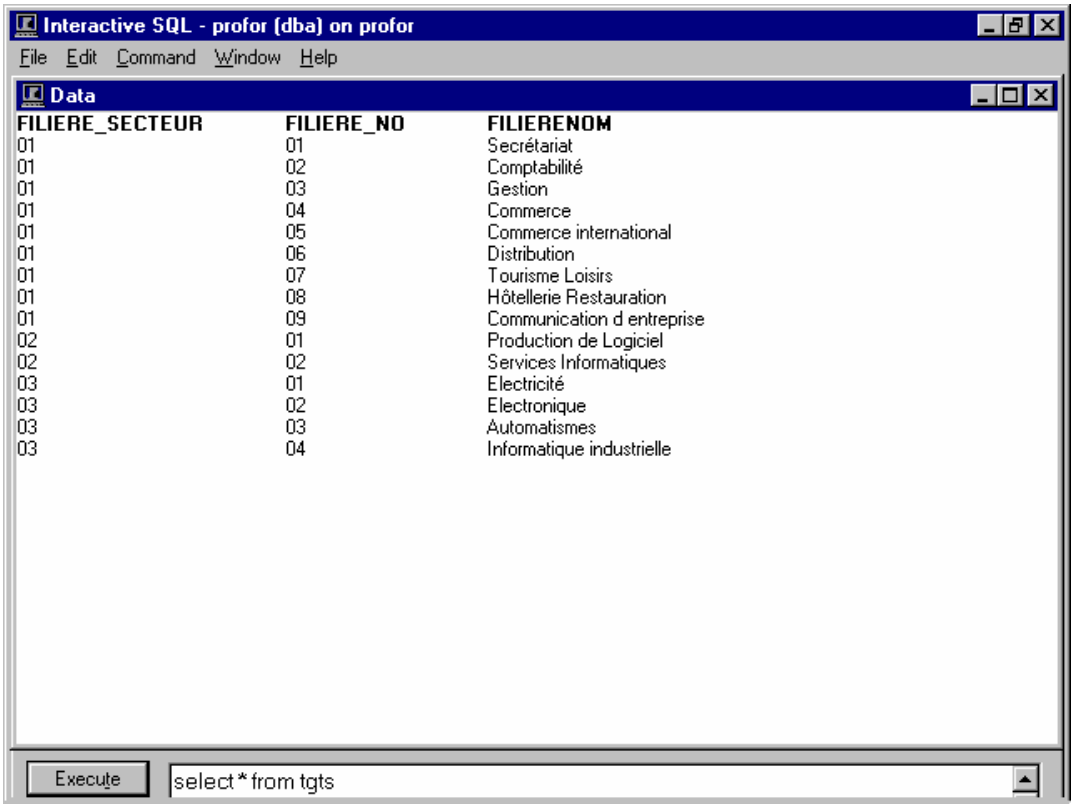




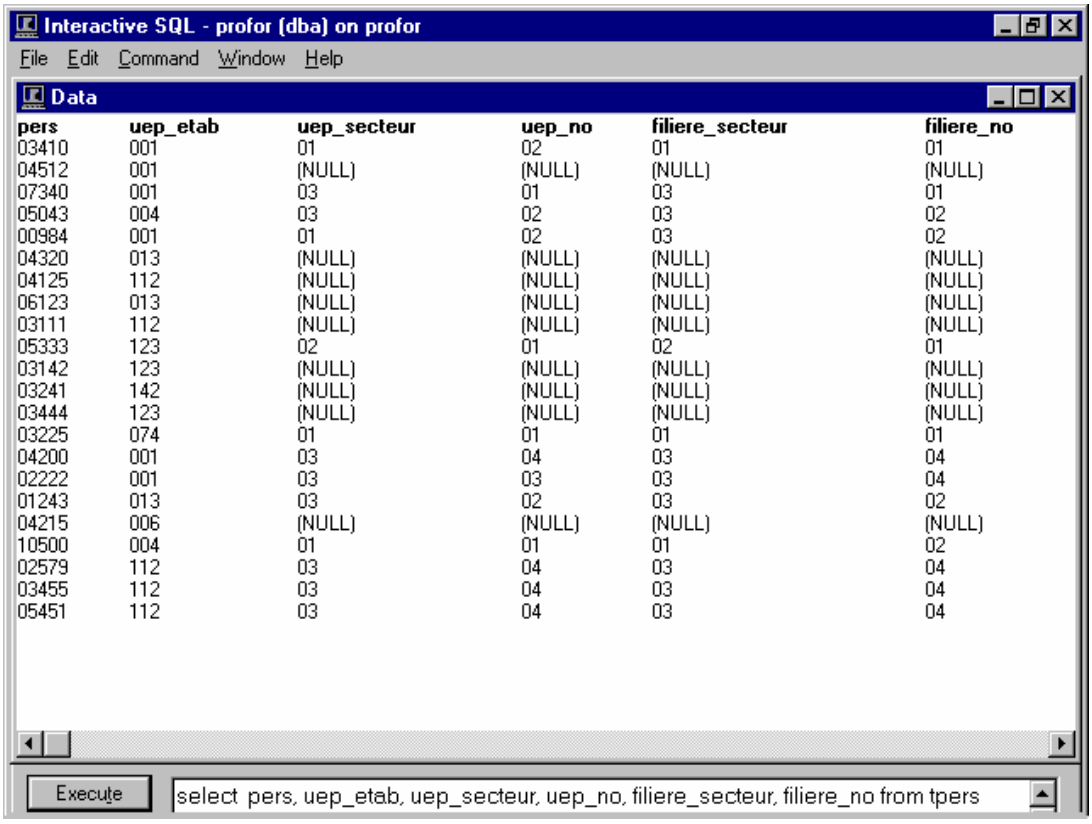




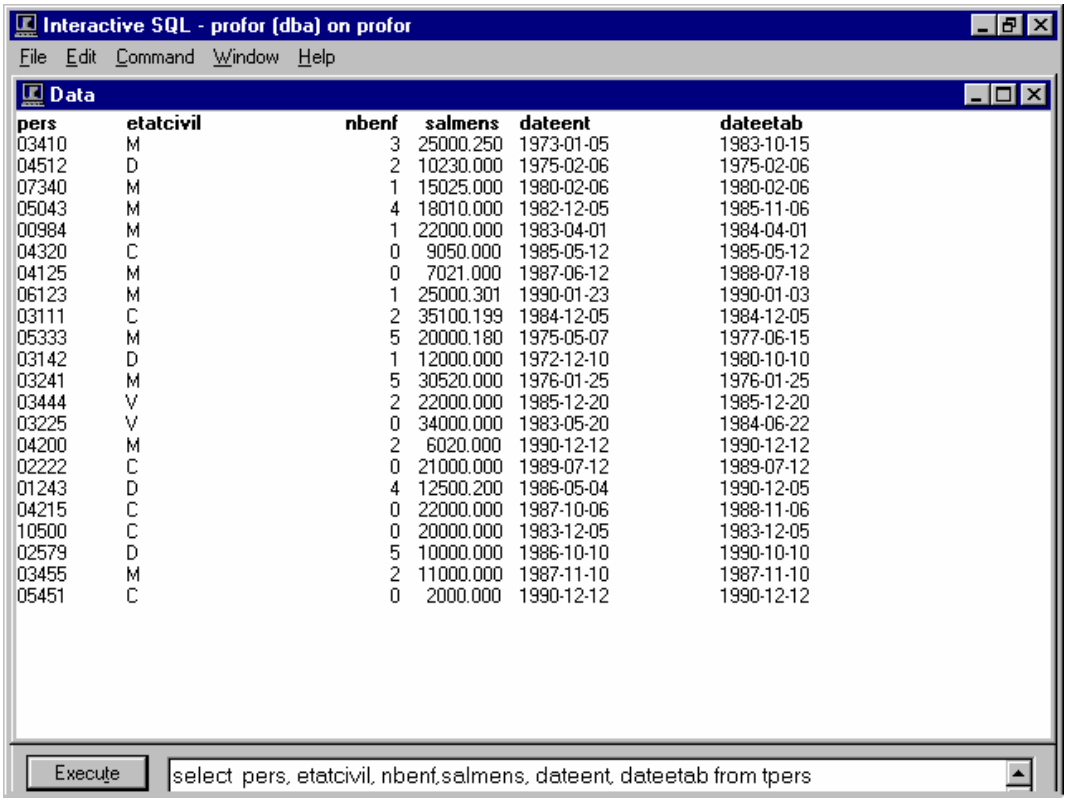




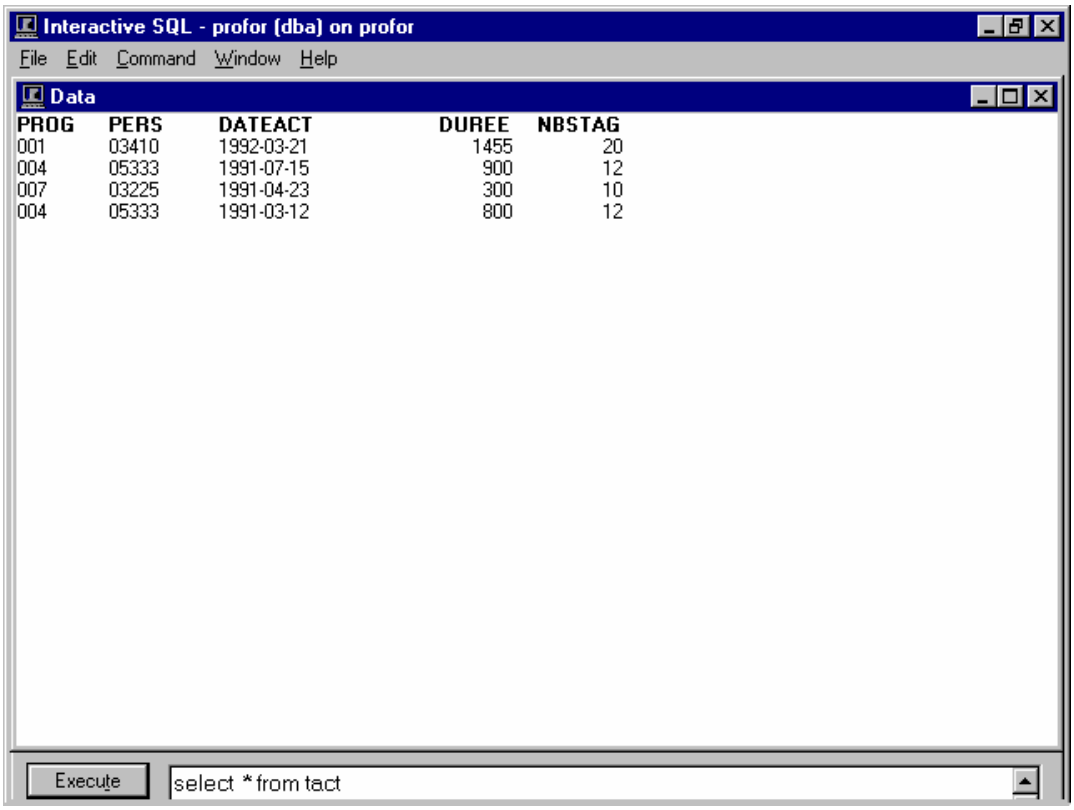
FILIERE_SECTEUR	FILIERE_NO	FILIERENOM
01	01	Secrétariat
01	02	Comptabilité
01	03	Gestion
01	04	Commerce
01	05	Commerce international
01	06	Distribution
01	07	Tourisme Loisirs
01	08	Hôtellerie Restauration
01	09	Communication d entreprise
02	01	Production de Logiciel
02	02	Services Informatiques
03	01	Electricité
03	02	Electronique
03	03	Automatismes
03	04	Informatique industrielle



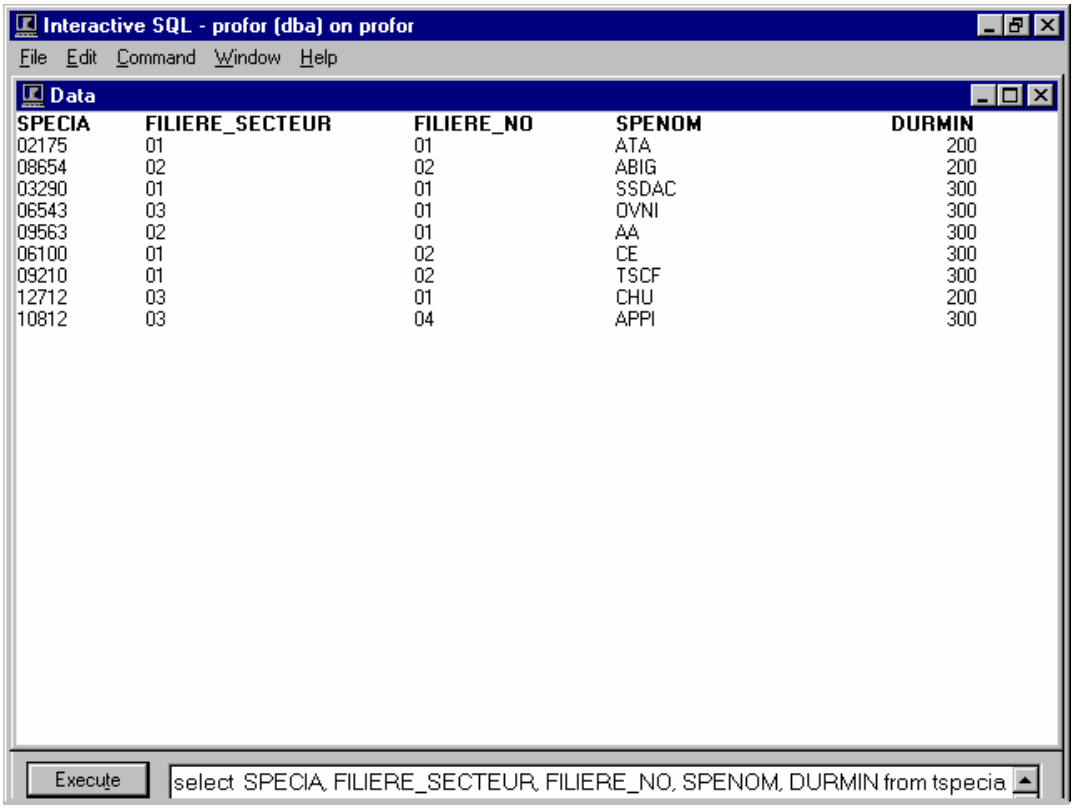
pers	uep_etab	uep_secteur	uep_no	filiere_secteur	filiere_no
03410	001	01	02	01	01
04512	001	(NULL)	(NULL)	(NULL)	(NULL)
07340	001	03	01	03	01
05043	004	03	02	03	02
00984	001	01	02	03	02
04320	013	(NULL)	(NULL)	(NULL)	(NULL)
04125	112	(NULL)	(NULL)	(NULL)	(NULL)
06123	013	(NULL)	(NULL)	(NULL)	(NULL)
03111	112	(NULL)	(NULL)	(NULL)	(NULL)
05333	123	02	01	02	01
03142	123	(NULL)	(NULL)	(NULL)	(NULL)
03241	142	(NULL)	(NULL)	(NULL)	(NULL)
03444	123	(NULL)	(NULL)	(NULL)	(NULL)
03225	074	01	01	01	01
04200	001	03	04	03	04
02222	001	03	03	03	04
01243	013	03	02	03	02
04215	006	(NULL)	(NULL)	(NULL)	(NULL)
10500	004	01	01	01	02
02579	112	03	04	03	04
03455	112	03	04	03	04
05451	112	03	04	03	04

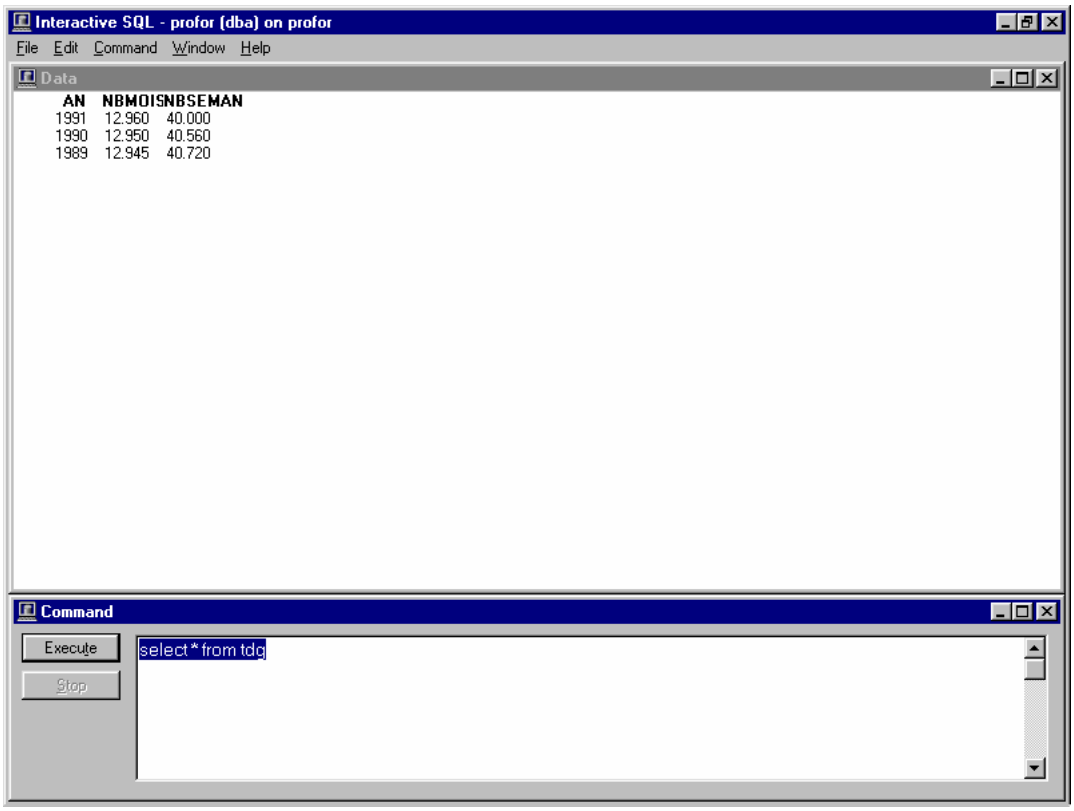


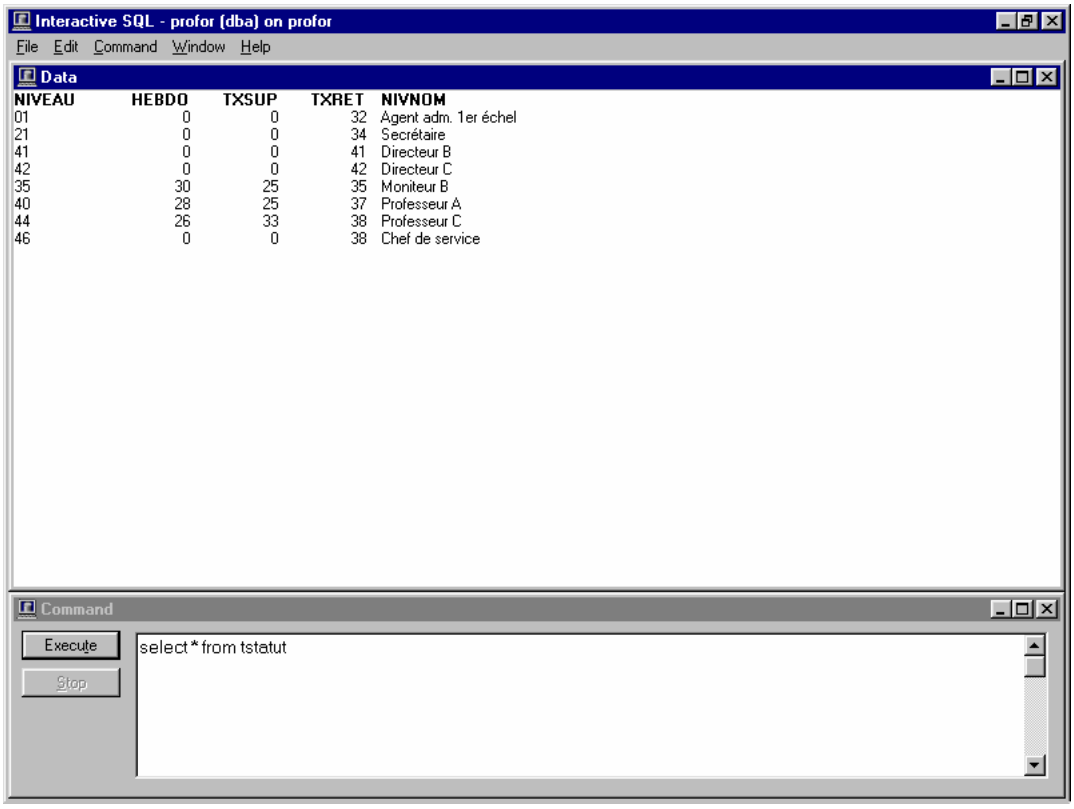
pers	etatcivil	nbenf	salmens	dateent	dateetab
03410	M	3	25000.250	1973-01-05	1983-10-15
04512	D	2	10230.000	1975-02-06	1975-02-06
07340	M	1	15025.000	1980-02-06	1980-02-06
05043	M	4	18010.000	1982-12-05	1985-11-06
00984	M	1	22000.000	1983-04-01	1984-04-01
04320	C	0	9050.000	1985-05-12	1985-05-12
04125	M	0	7021.000	1987-06-12	1988-07-18
06123	M	1	25000.301	1990-01-23	1990-01-03
03111	C	2	35100.199	1984-12-05	1984-12-05
05333	M	5	20000.180	1975-05-07	1977-06-15
03142	D	1	12000.000	1972-12-10	1980-10-10
03241	M	5	30520.000	1976-01-25	1976-01-25
03444	V	2	22000.000	1985-12-20	1985-12-20
03225	V	0	34000.000	1983-05-20	1984-06-22
04200	M	2	6020.000	1990-12-12	1990-12-12
02222	C	0	21000.000	1989-07-12	1989-07-12
01243	D	4	12500.200	1986-05-04	1990-12-05
04215	C	0	22000.000	1987-10-06	1988-11-06
10500	C	0	20000.000	1983-12-05	1983-12-05
02579	D	5	10000.000	1986-10-10	1990-10-10
03455	M	2	11000.000	1987-11-10	1987-11-10
05451	C	0	2000.000	1990-12-12	1990-12-12

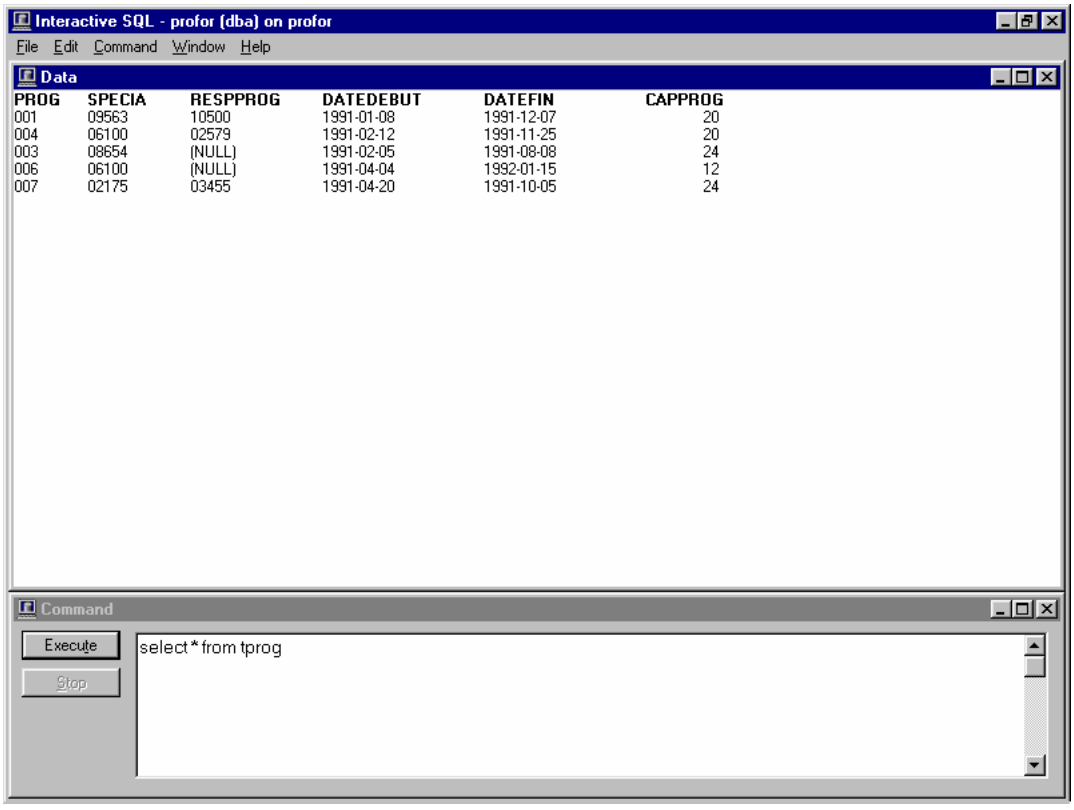


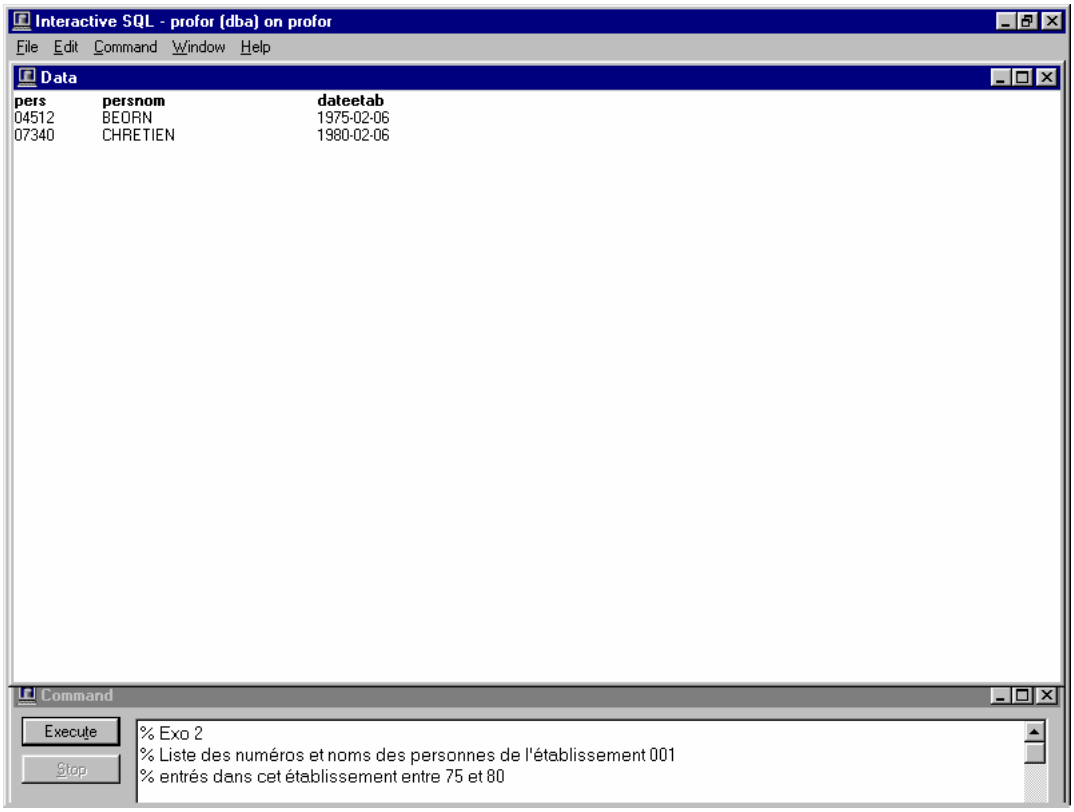
PROG	PERS	DATEACT	DUREE	NBSTAG
001	03410	1992-03-21	1455	20
004	05333	1991-07-15	900	12
007	03225	1991-04-23	300	10
004	05333	1991-03-12	800	12

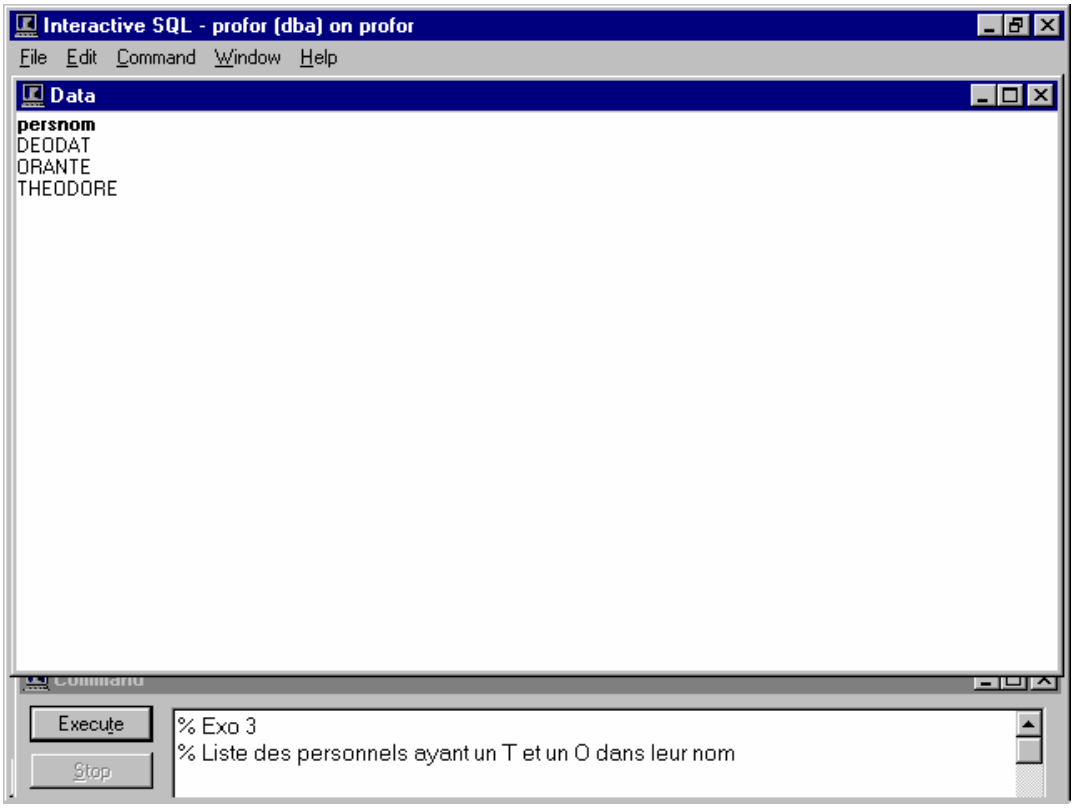


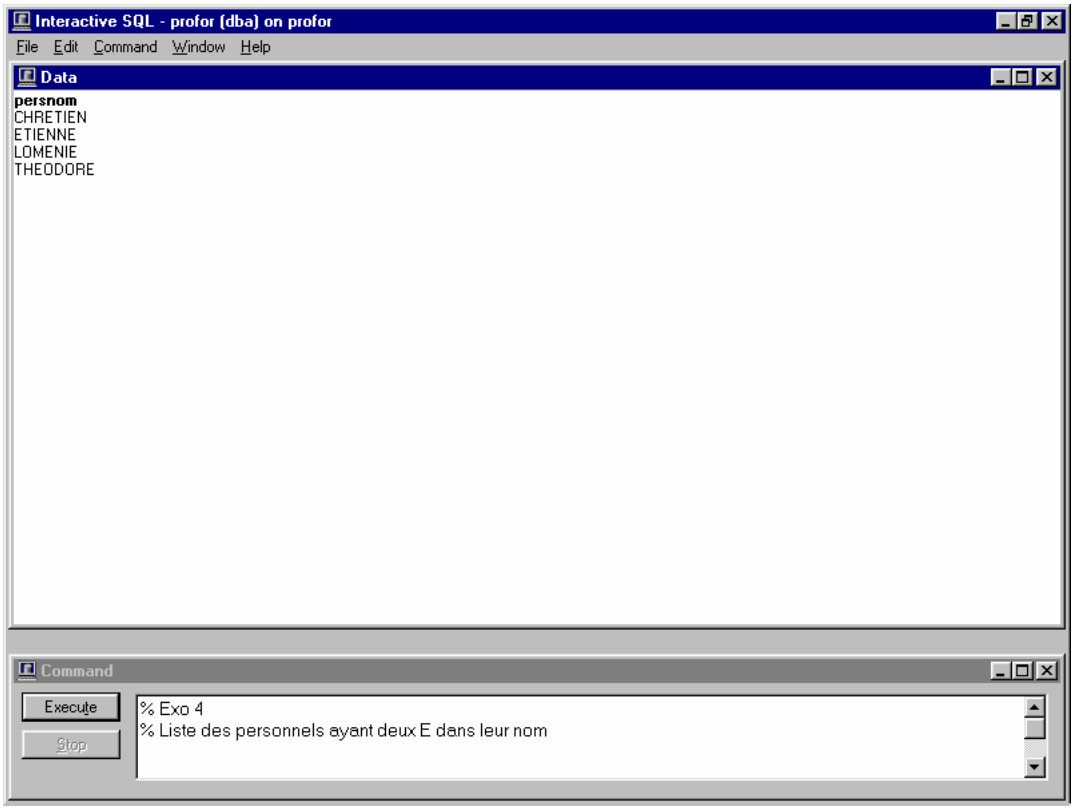


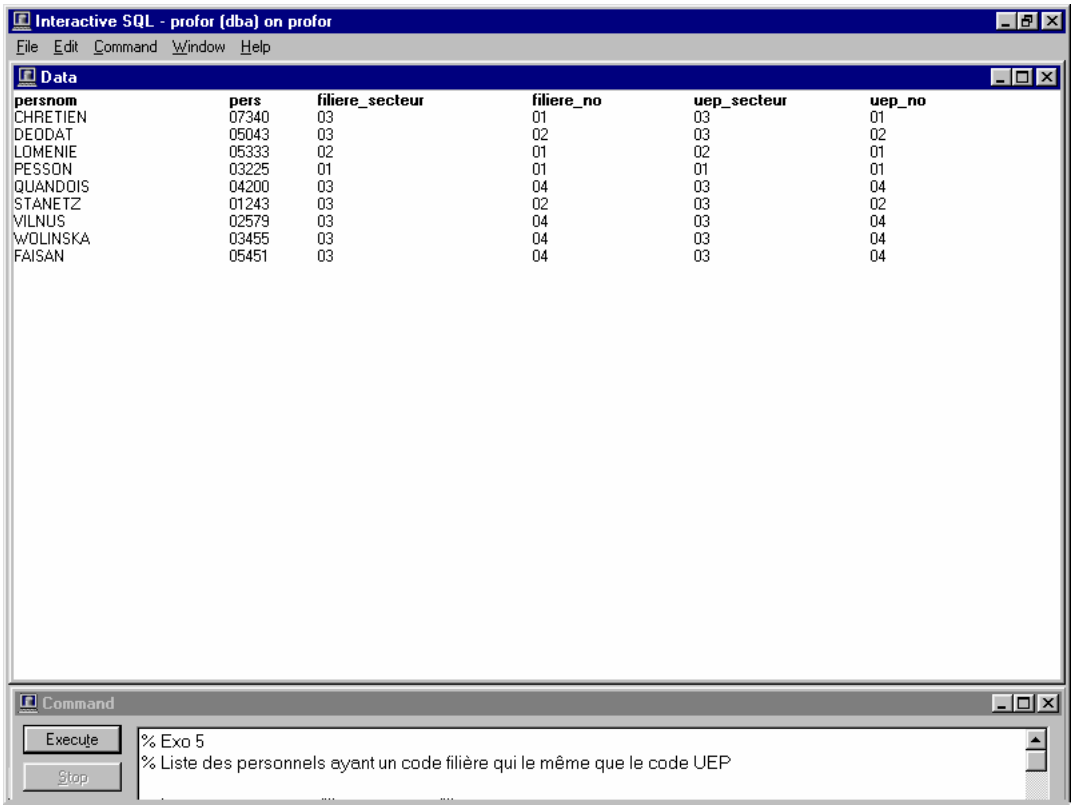


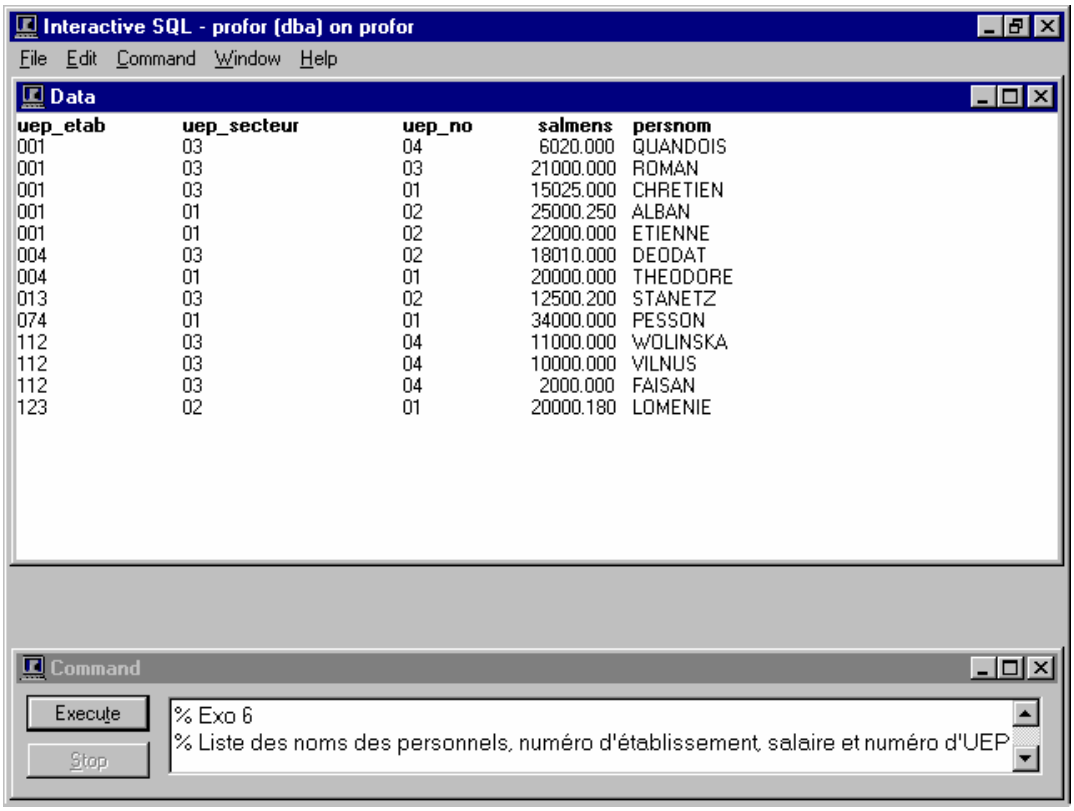


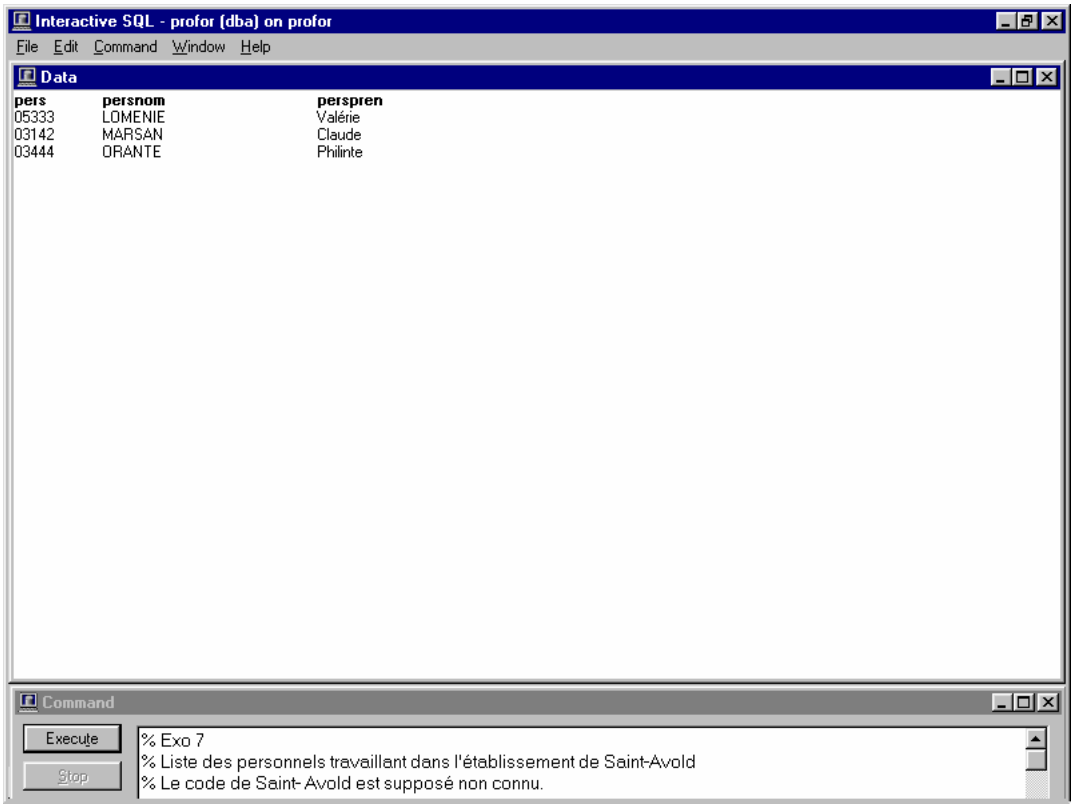


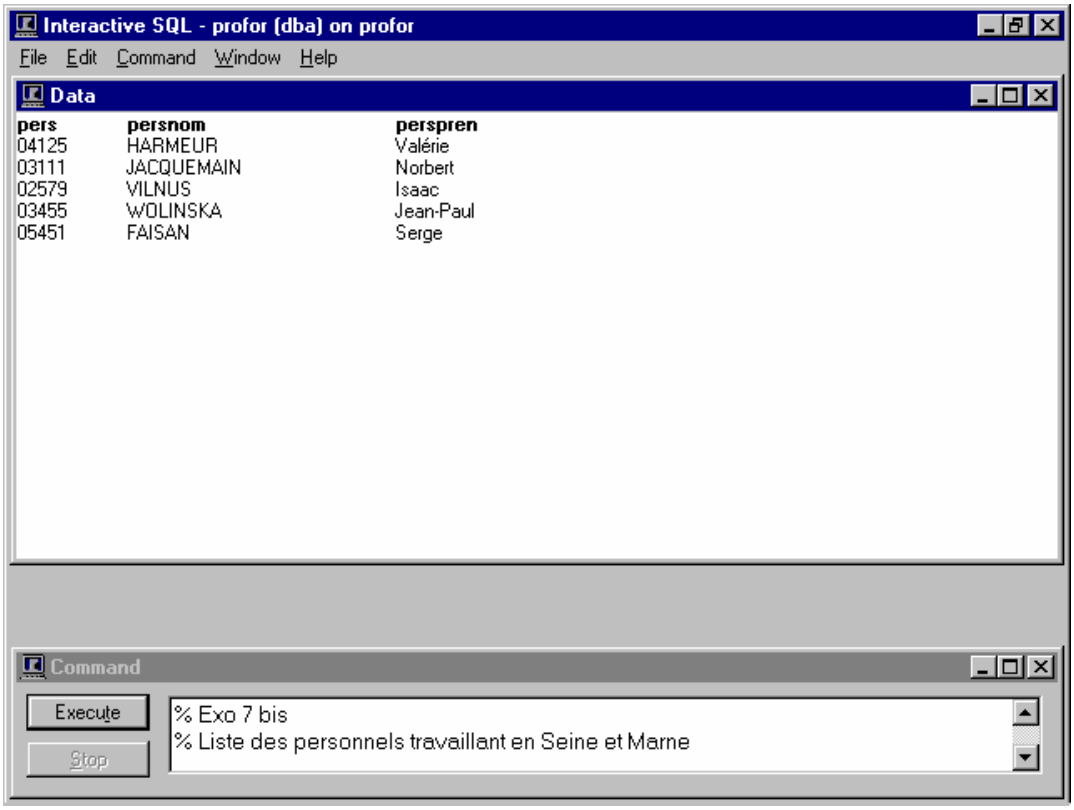


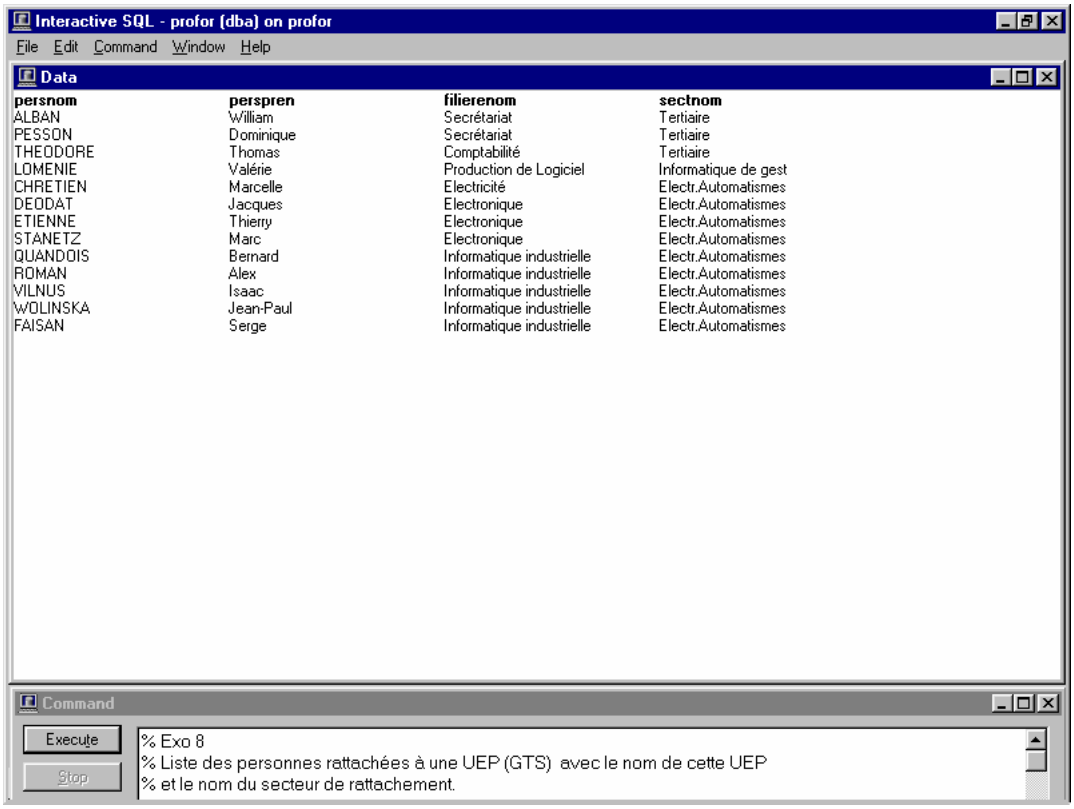


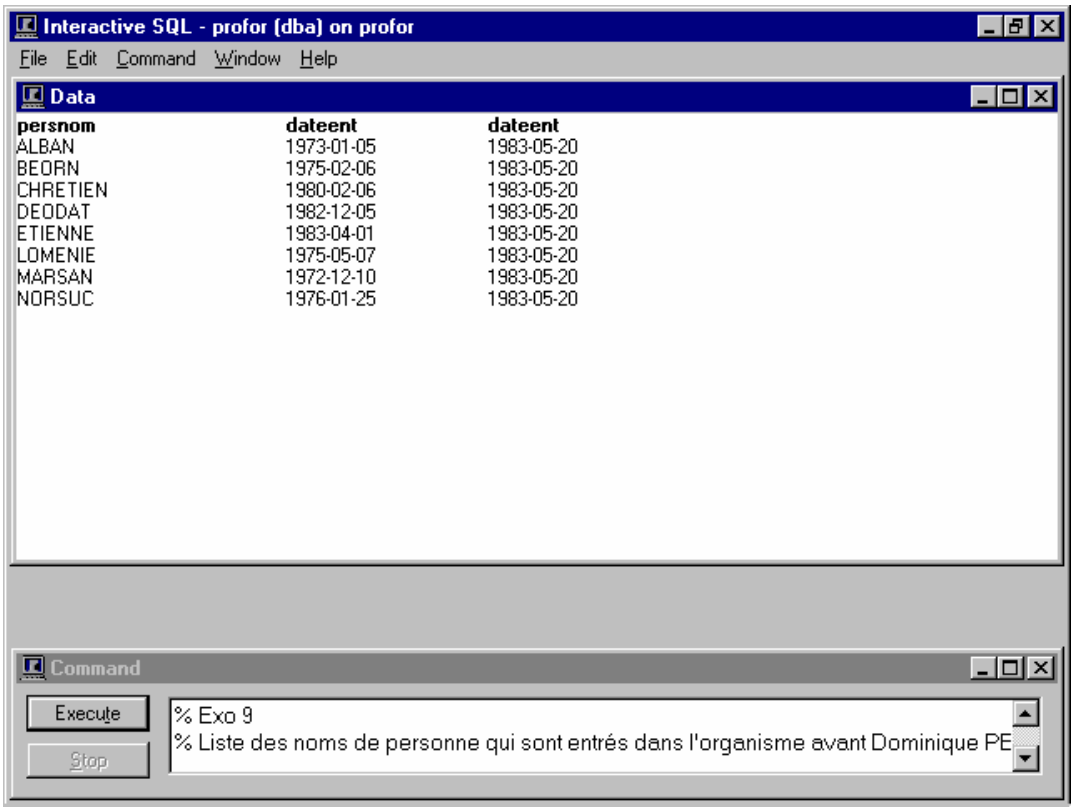


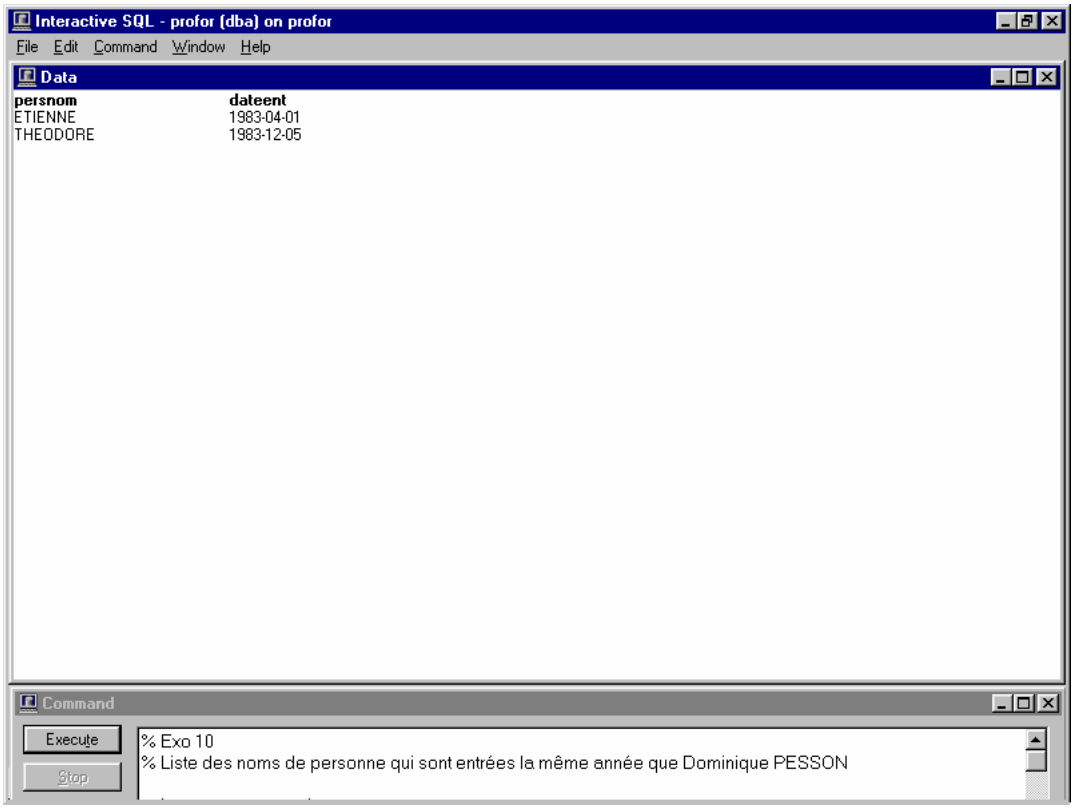


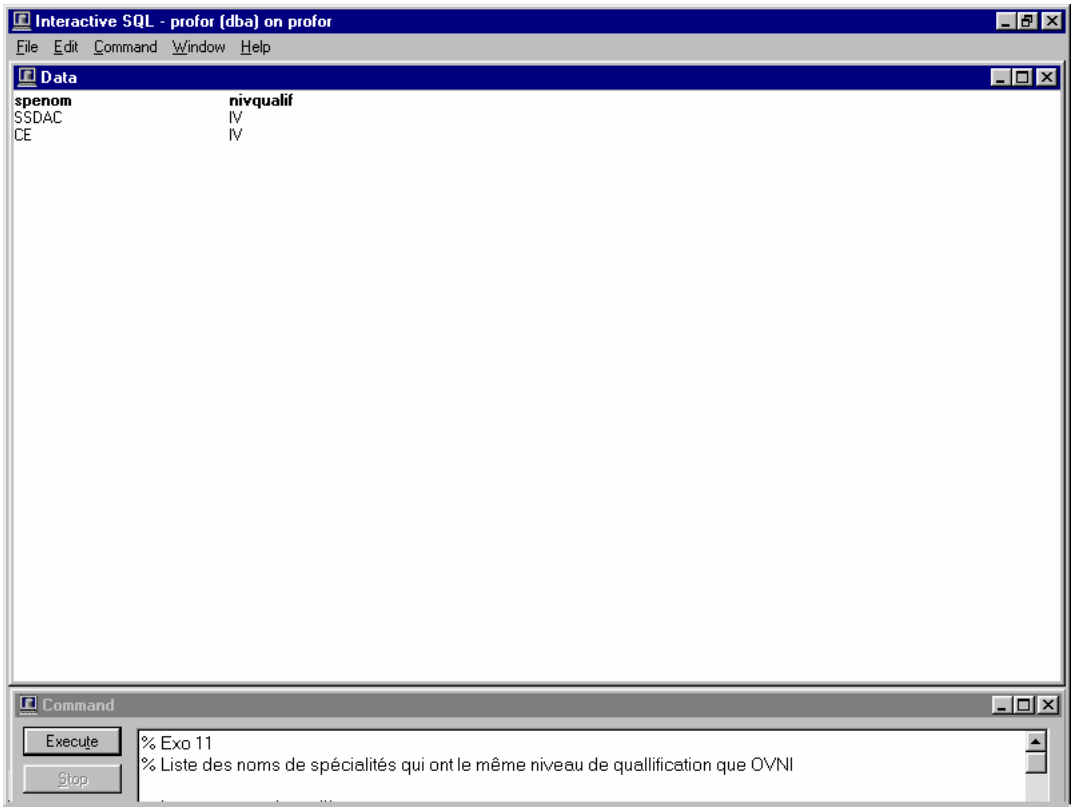


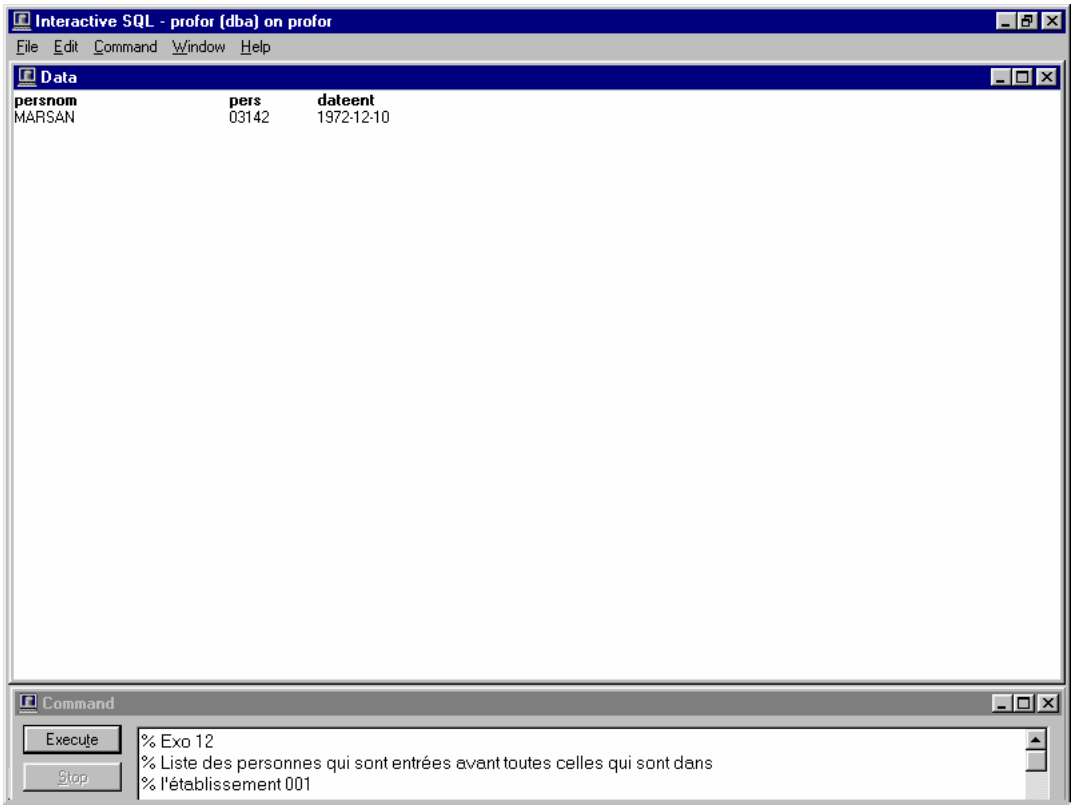


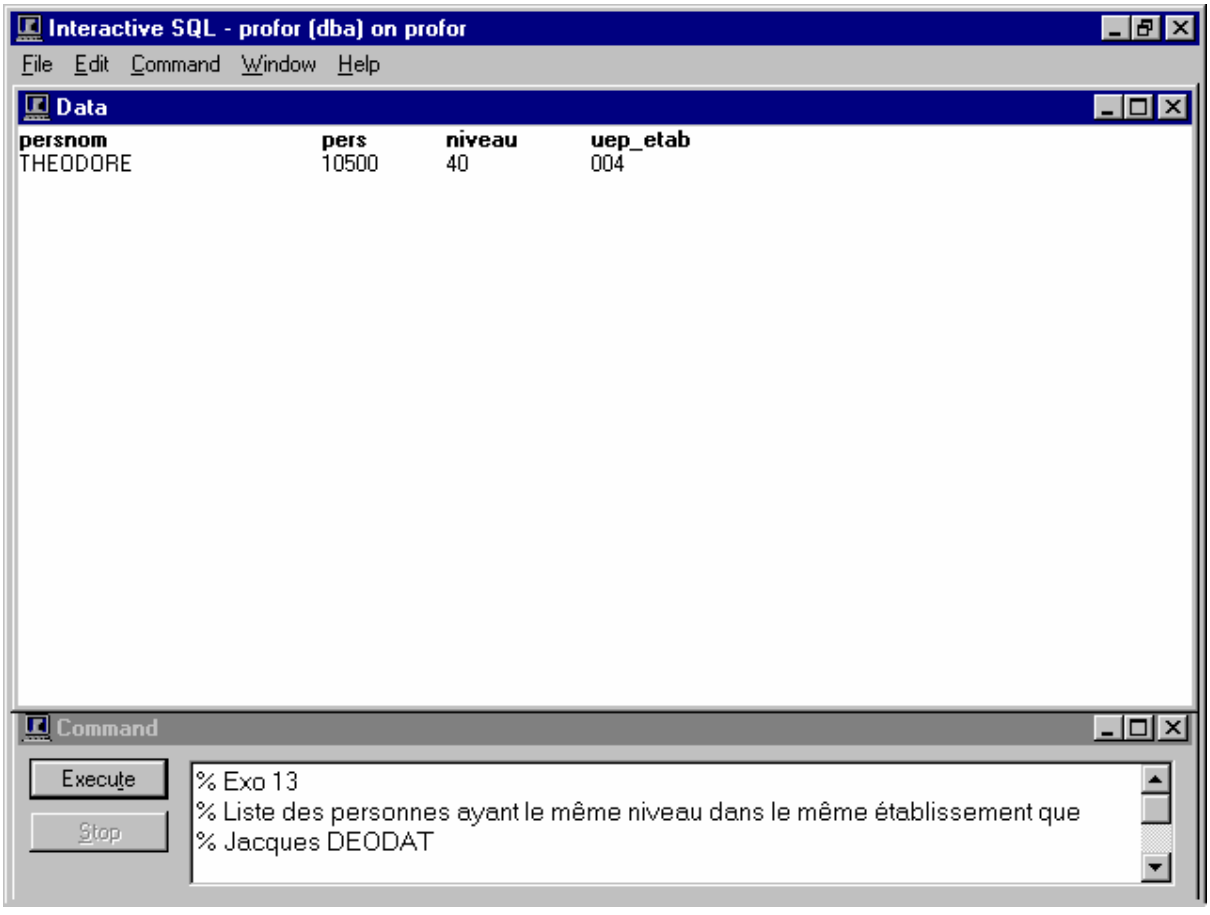


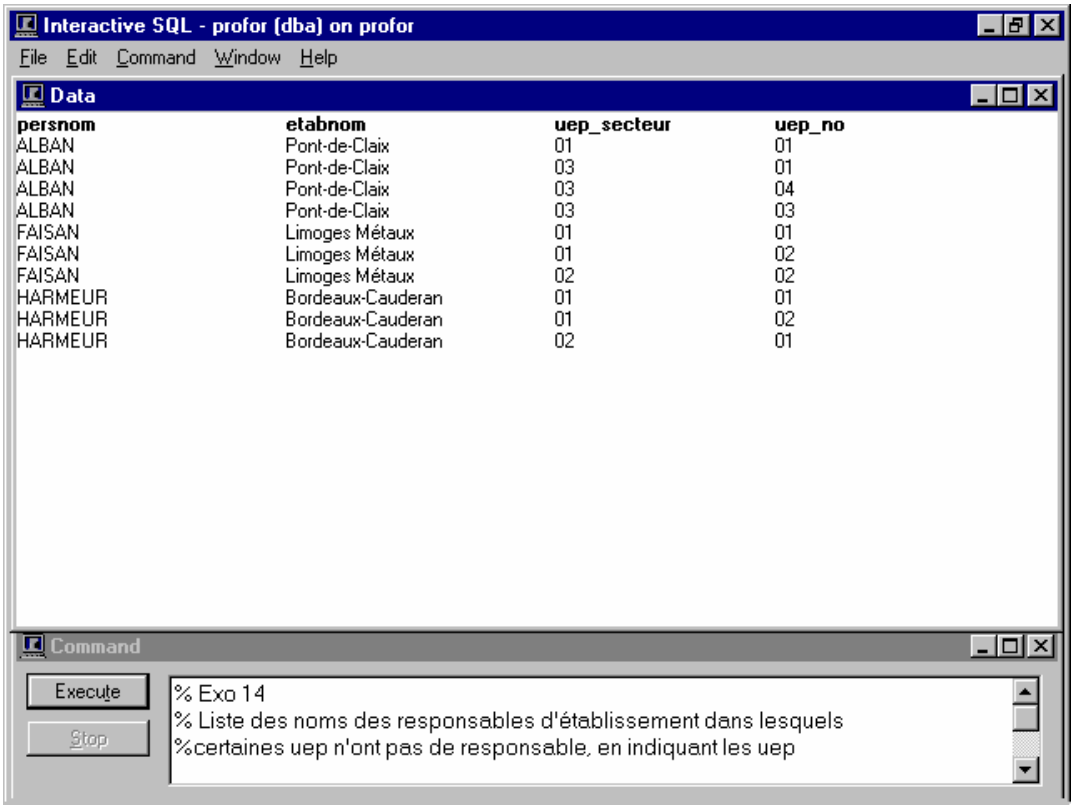


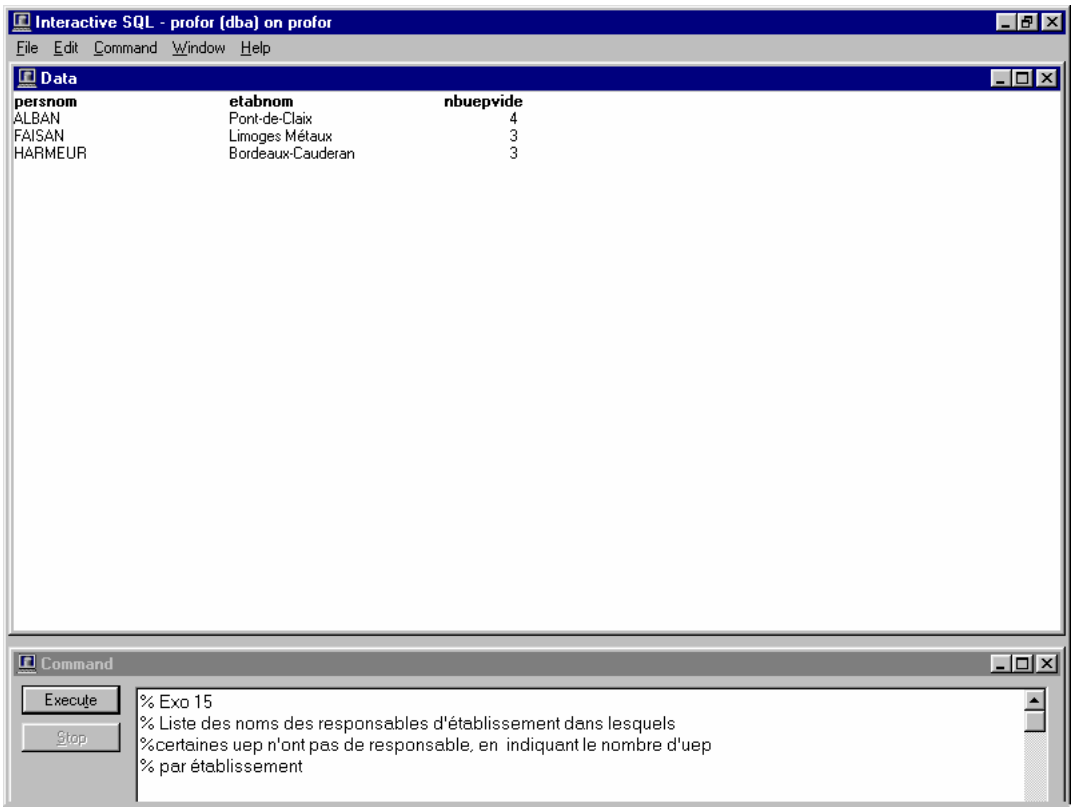


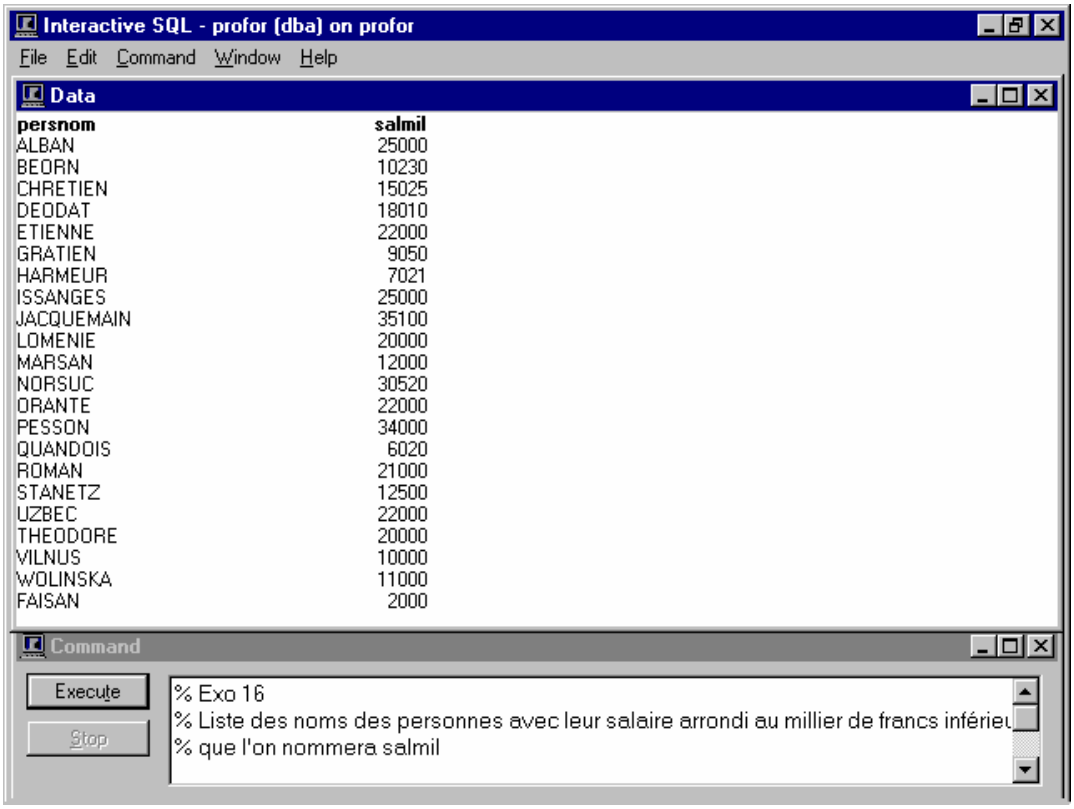


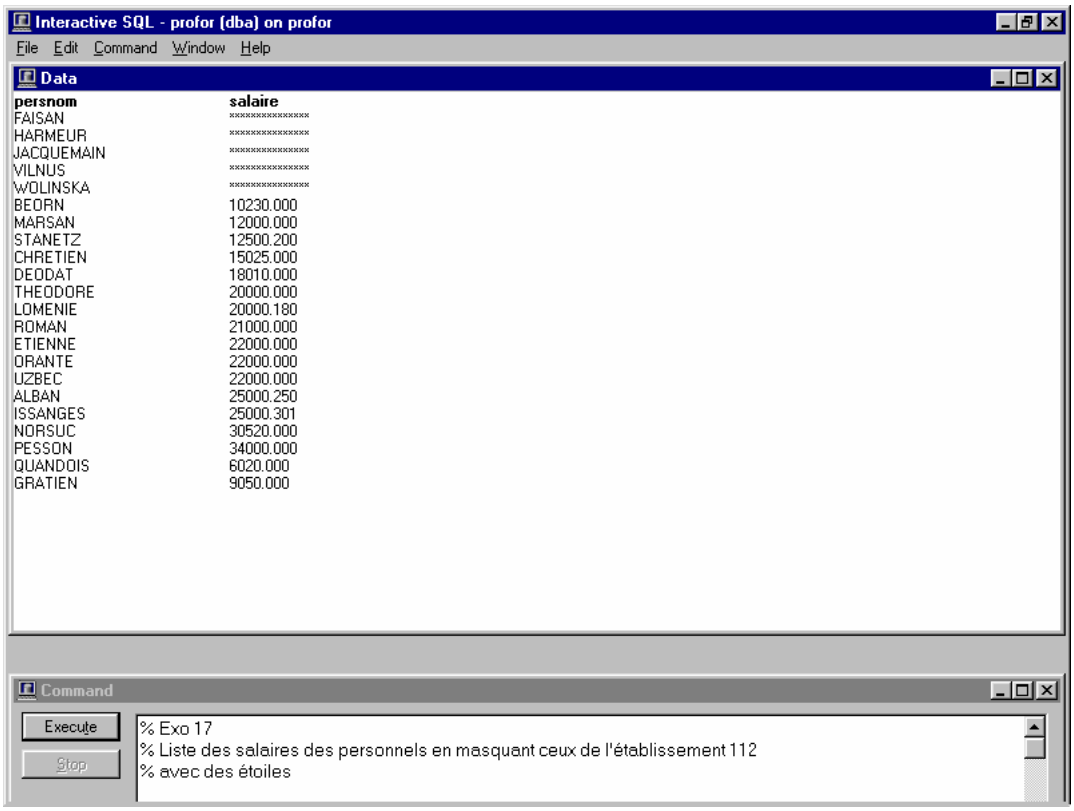


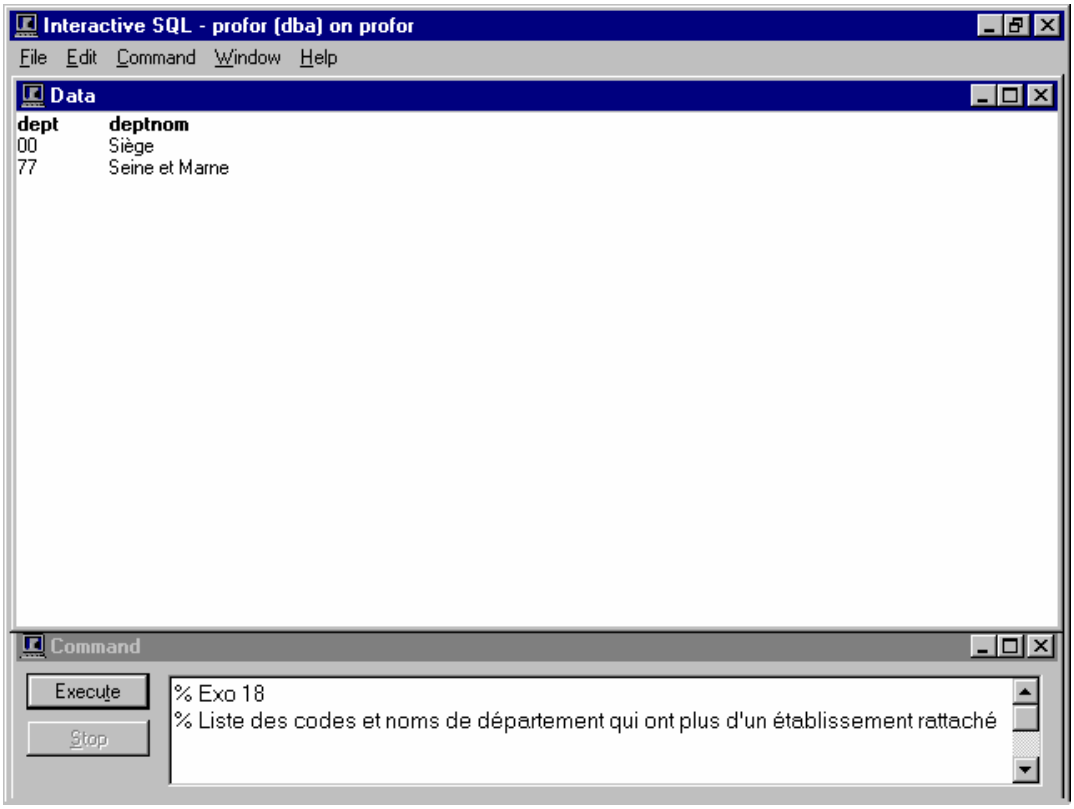


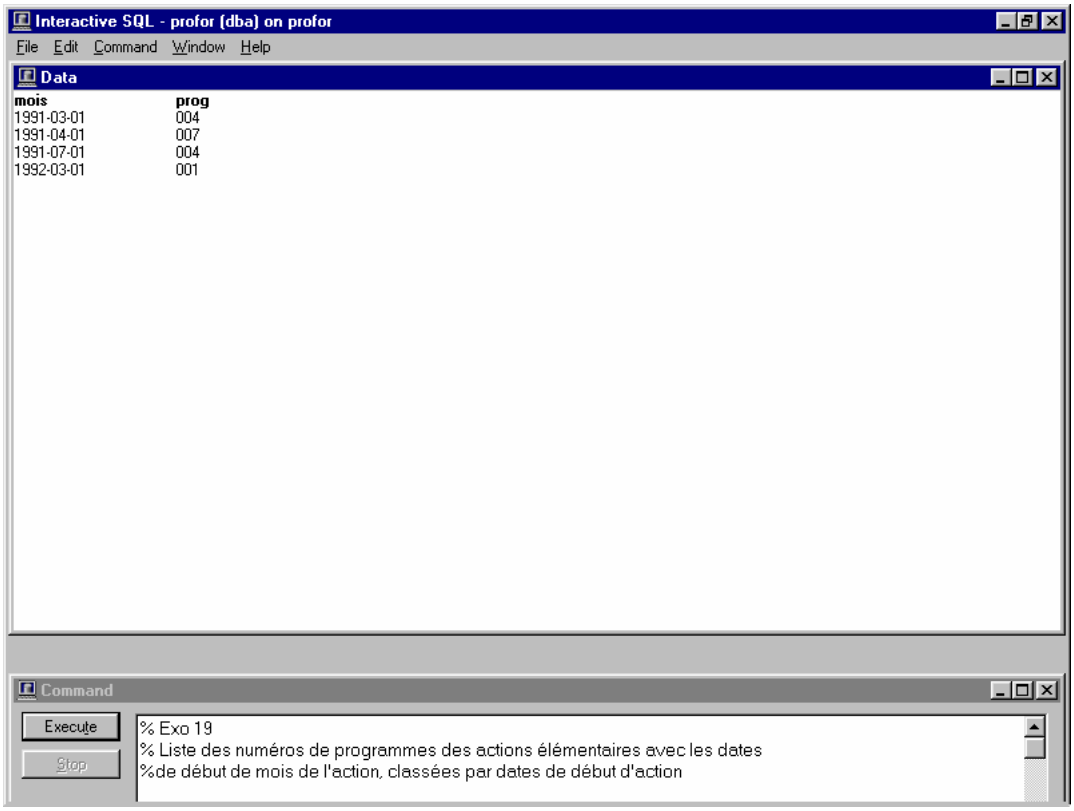


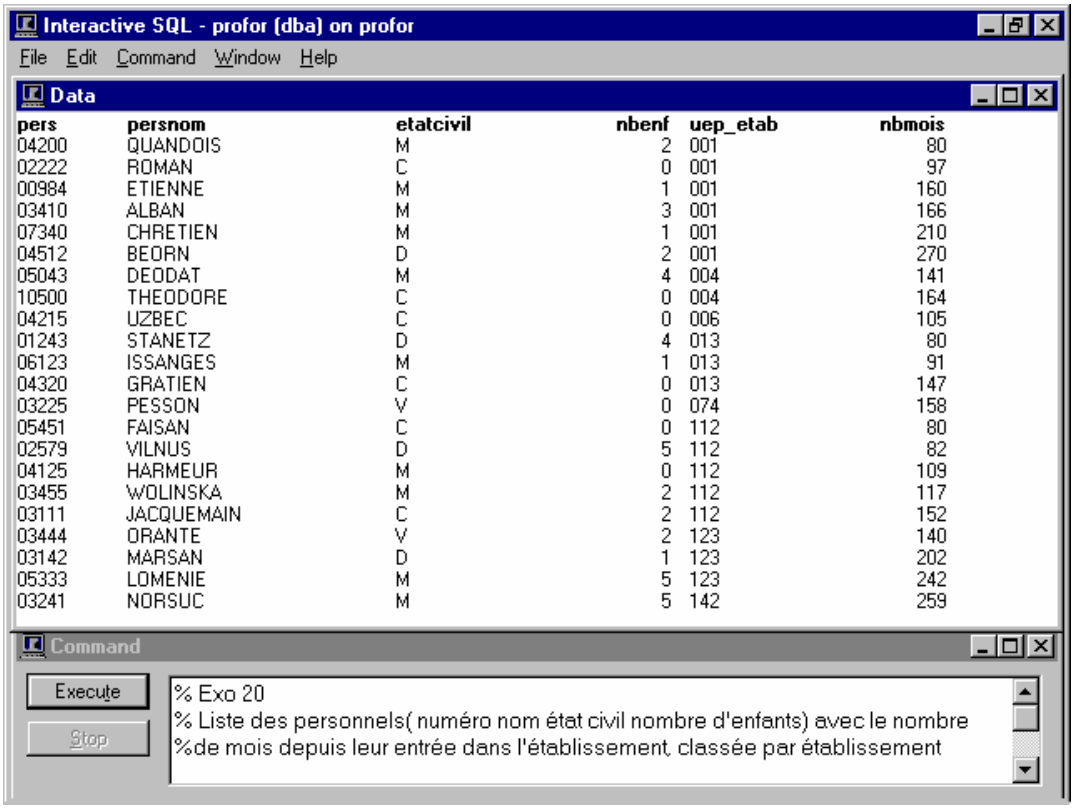


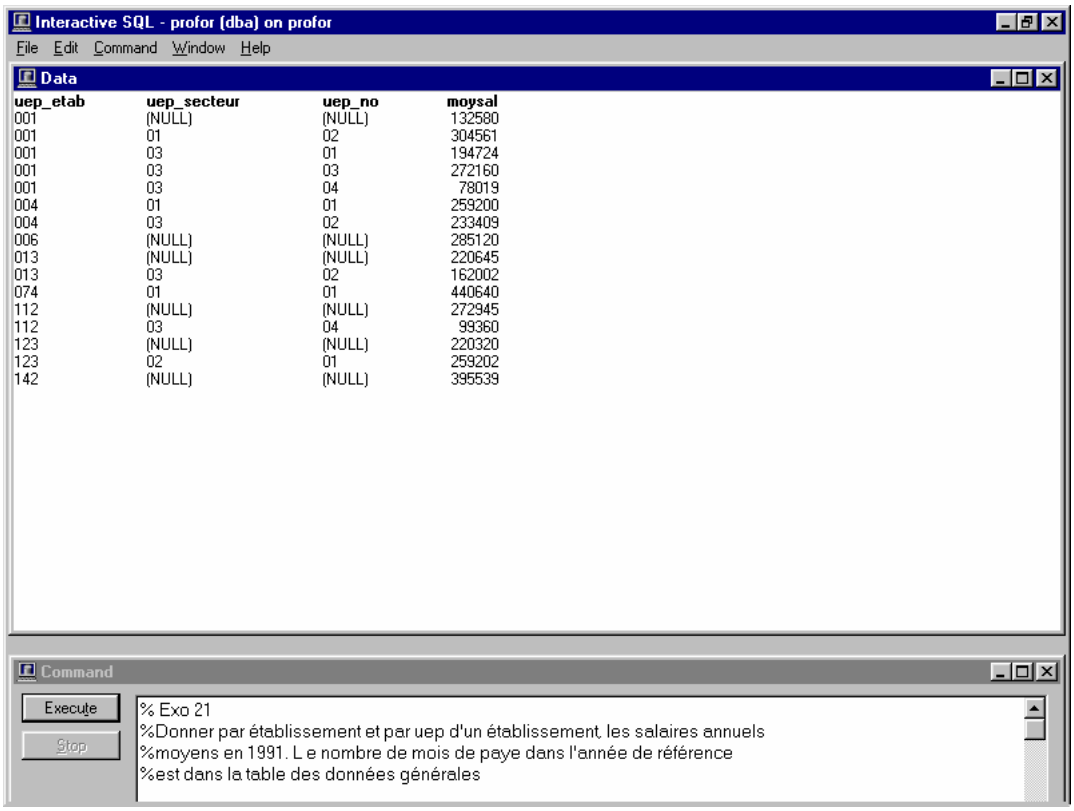


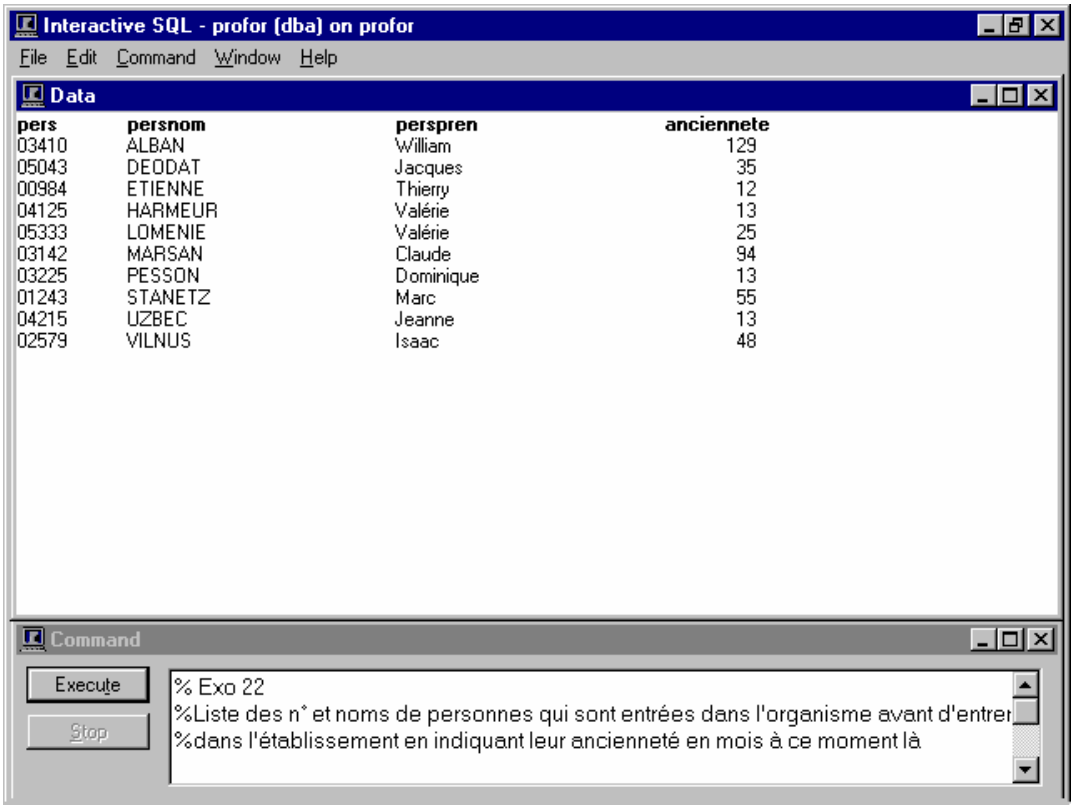


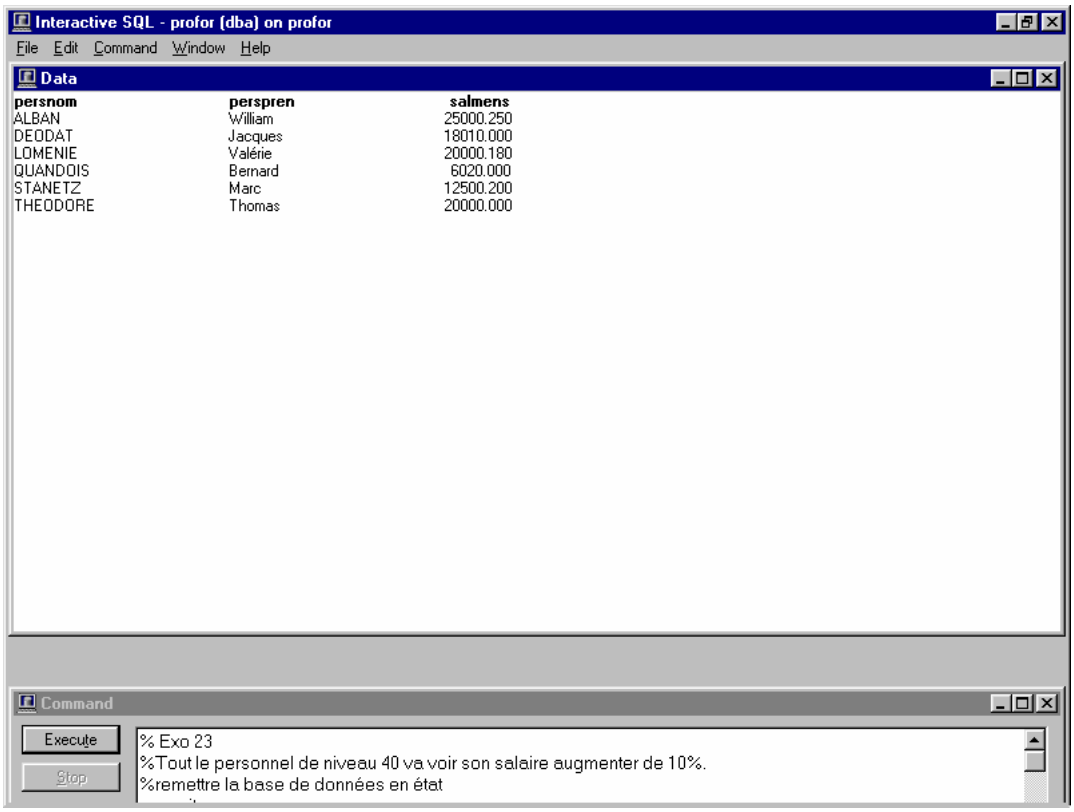


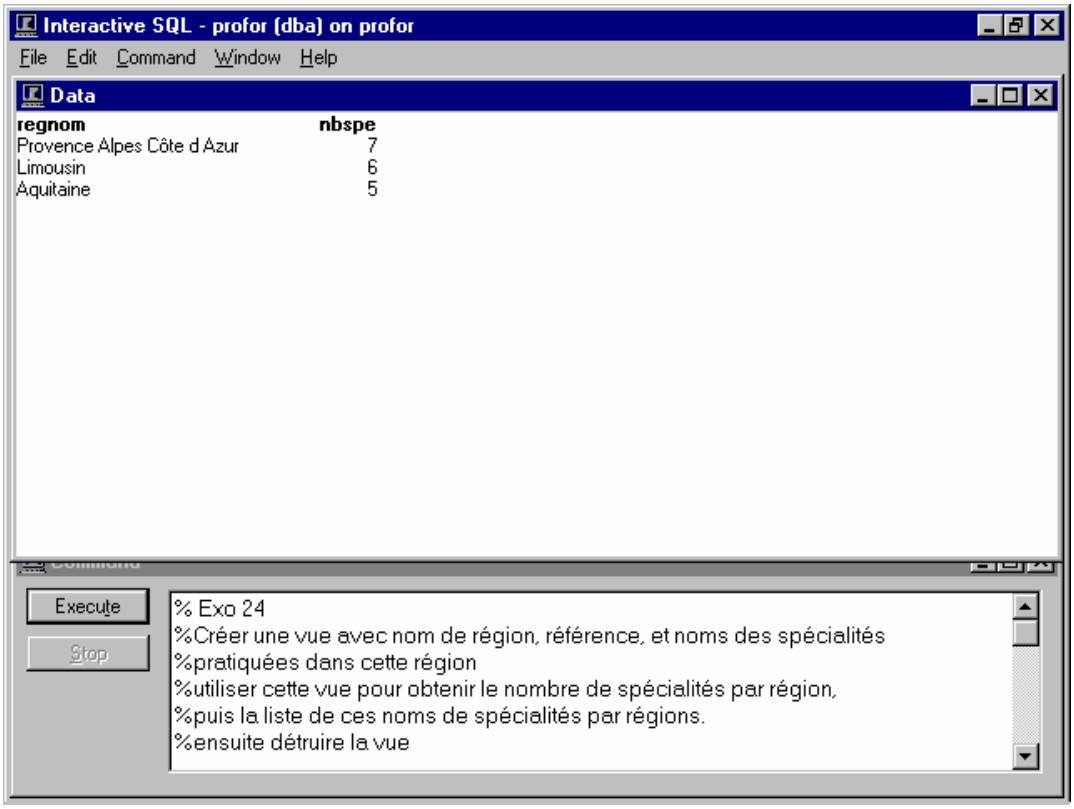


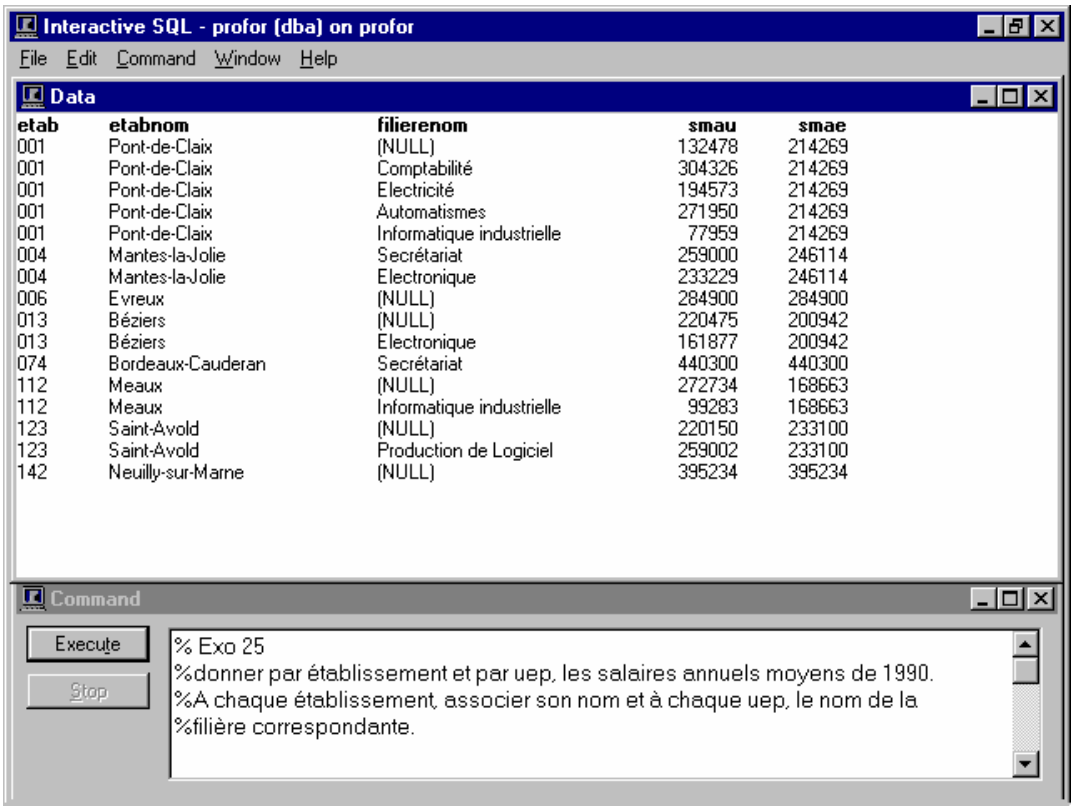


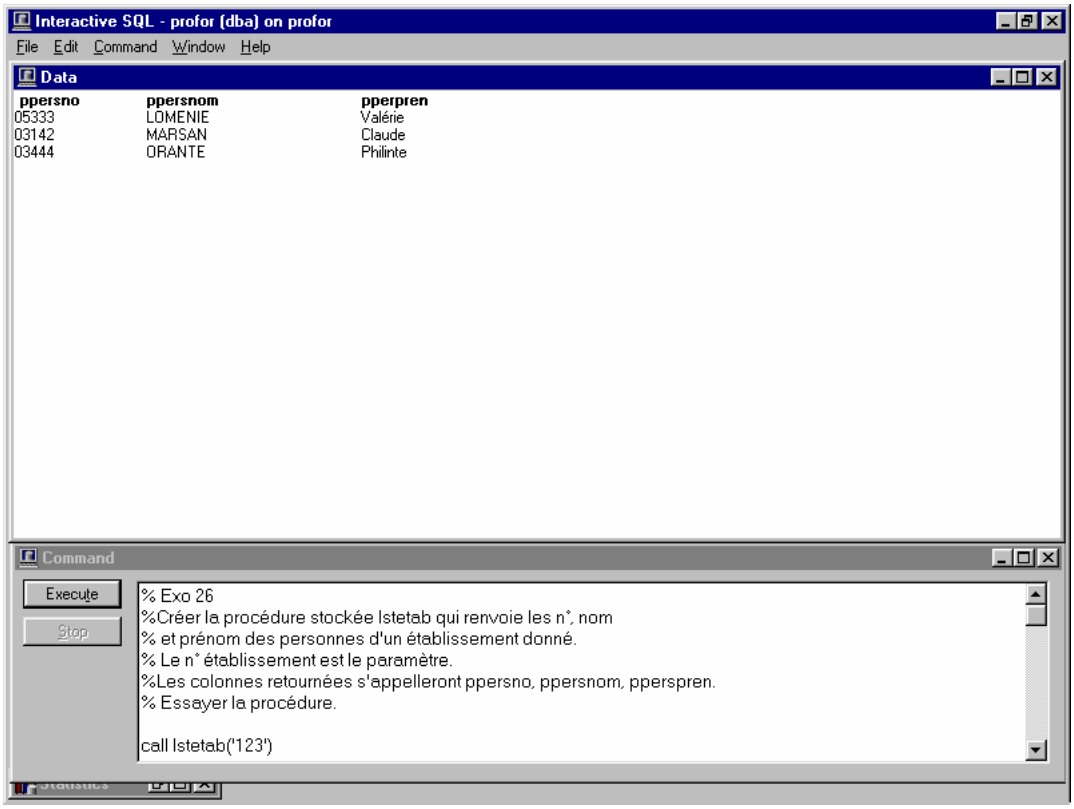


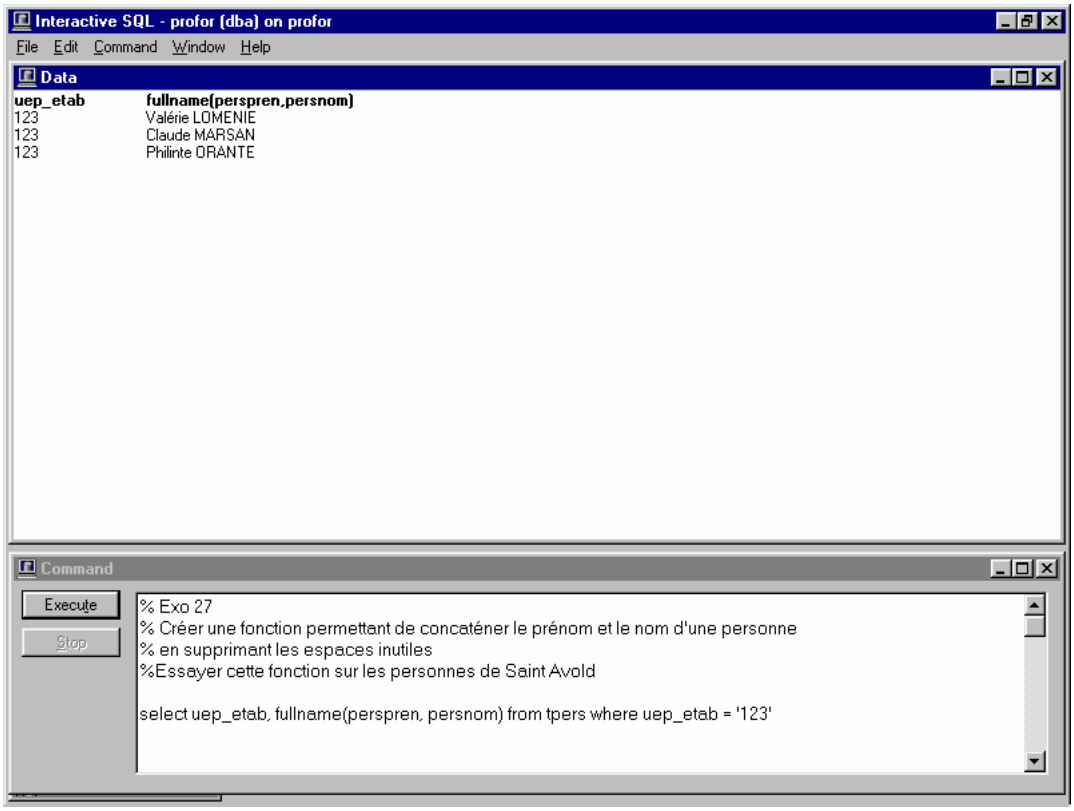


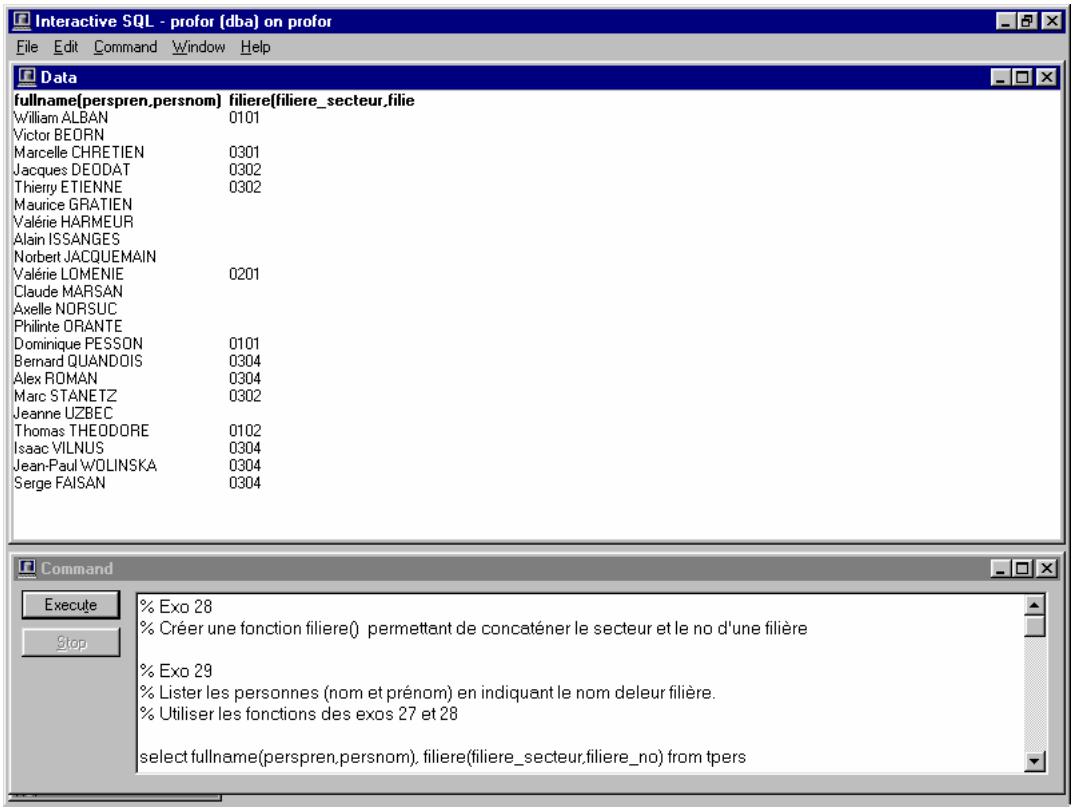


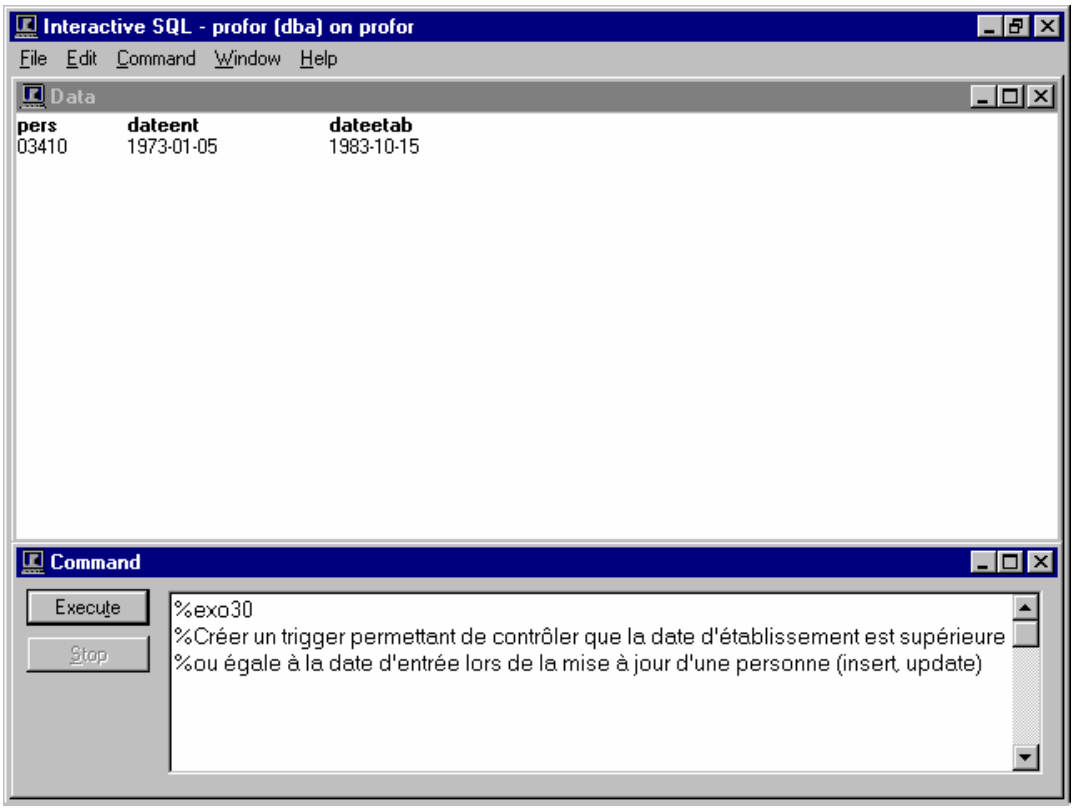


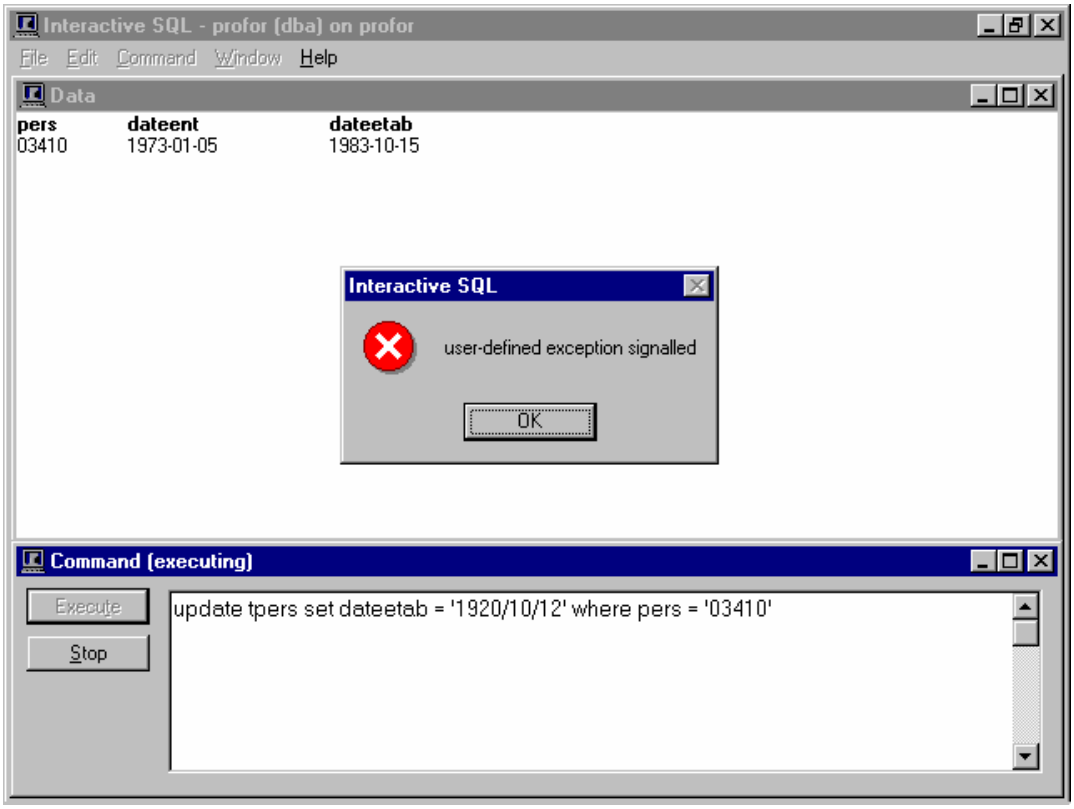












4 - Traitements dans les BDD

1 - Procédures

Une procédure stockée ou procédure mémorisée ou stored procedure est un ensemble d'instructions SQL précompilées et mémorisées dans le dictionnaire.
Elle sera exécutée à chaque fois qu'on l'appelle.

Création d'une procédure :

```
CREATE PROCEDURE new_dpt
  (IN id INT, IN nom CHAR(35), IN reg INT)
BEGIN
  INSERT INTO departement(dept_id, dept_nom, dept_reg)
            VALUES(id, nom, reg) ;
END
```

Appel de la procédure :

```
CALL new_dpt (59, 'Nord', 15) ;
```

Suppression de la procédure :

```
DROP PROCEDURE new_dpt ;
```

Procédure retournant un résultat paramétré:

```
CREATE PROCEDURE SalaireMoyen
  (OUT salmoy DECIMAL(12,2))
BEGIN
  SELECT AVG(salaire) INTO salmoy FROM employé ;
END
```

Appel de la procédure :

```
CREATE VARIABLE moyenne NUMERIC(12,2)
CALL SalaireMoyen  (moyenne) ;
```

Procédure paramètrée retournant une série :

```
CREATE PROCEDURE ListeSalaire
  (IN dpt INT)
  RESULT (« Numéro » INT, « Salaire » NUMERIC(12,2))
BEGIN
  SELECT emp_num, emp_salaire
    FROM employé
    WHERE emp_dept = dpt;
END
```

Appel de la procédure :

```
CALL ListeSalaire (59) ;
```

4 - Traitements dans les BDD

2 - Fonctions

Création d'une fonction :

```
CREATE FUNCTION nomComplet
  (prenom CHAR(20), nom CHAR(20))
RETURNS CHAR(41)
BEGIN
  DECLARE varNom CHAR(41) ;
  SET varNom = prenom || ' ' || nom ;
  RETURN (varNom) ;
END
```

Utilisation de la fonction :

```
SELECT nomComplet(emp_prenom, emp_nom)
  FROM employe ;
```

Suppression de la fonction :

```
DROP FUNCTION nomComplet ;
```

4 - Traitements dans les BDD

3 - Triggers

Un trigger est une procédure compilée, cataloguée dans le dictionnaire, qui s'exécute automatiquement chaque fois que l'événement déclenchant associé se produit : before update, after update, before insert, after insert, before delete, after delete.

Création d'un trigger :

```
CREATE TRIGGER controleDate
   AFTER INSERT, UPDATE ON employe
   REFERENCING NEW AS new_employe
   FOR EACH STATEMENT
BEGIN
   DECLARE erreur EXCEPTION FOR SQLSTATE '99999' ;
   IF new_employe.dateNais > now THEN
      SIGNAL erreur ;
   END IF ;

EXCEPTION
   WHEN erreur THEN
            MESSAGE 'Date naissance supérieure à date du jour !' ;
            SIGNAL erreur ;
   WHEN OTHERS THEN
            MESSAGE 'Problème de maj de la table employe' ;
            RESIGNAL ;
END
```

Exécution du Trigger :

```
INSERT INTO employe (…, dateNais, ….)
   VALUES (…) ;

UPDATE employe
   SET dateNais = … ;
```

Suppression du Trigger :

```
DROP TRIGGER controleDate
```

6 - Accès aux BDD dans les langages hôtes

Techniques abordées :

- ODBC
- SGBD : SQL Server (Windows NT)
- Visual Basic : contrôle ADODC
- Transaction : BeginTrans, CommitTrans, RollBack
- ordres SQL : Select, Insert, Update, Delete

Réalisations :

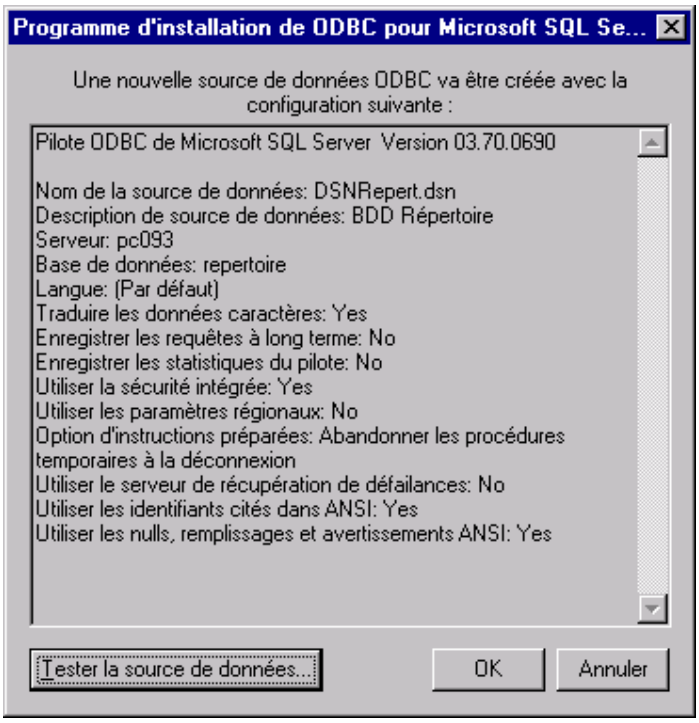
Réaliser une application permettant de gérer un fichier répertoire sur SQL Server via OLEDB.

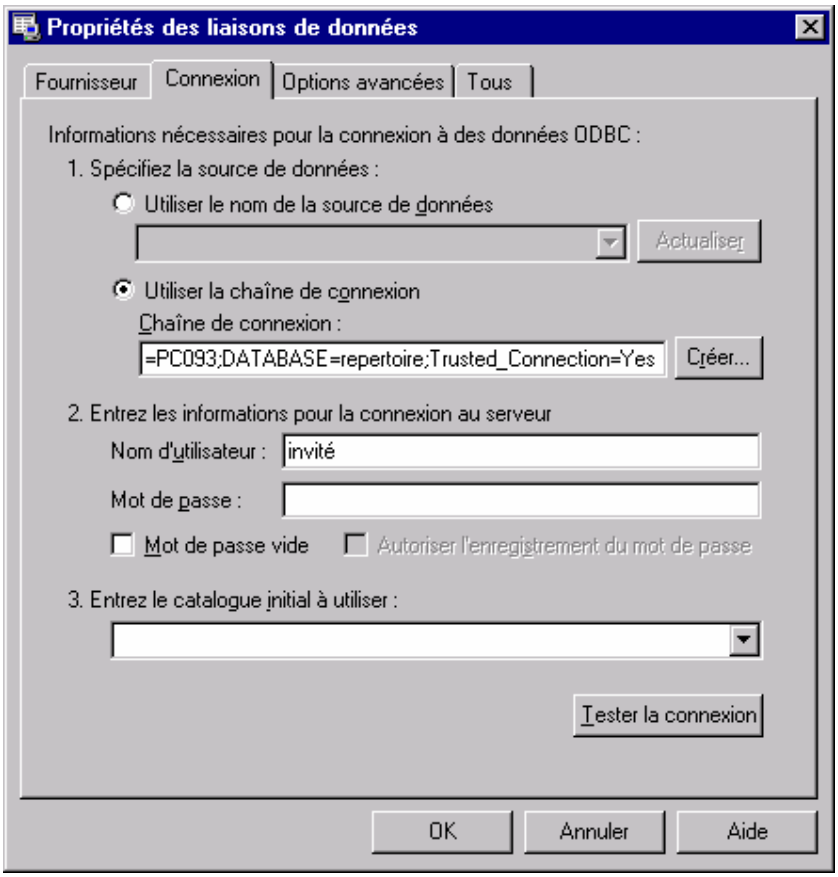
- Structure de la Base De Données "**repertoire**" :

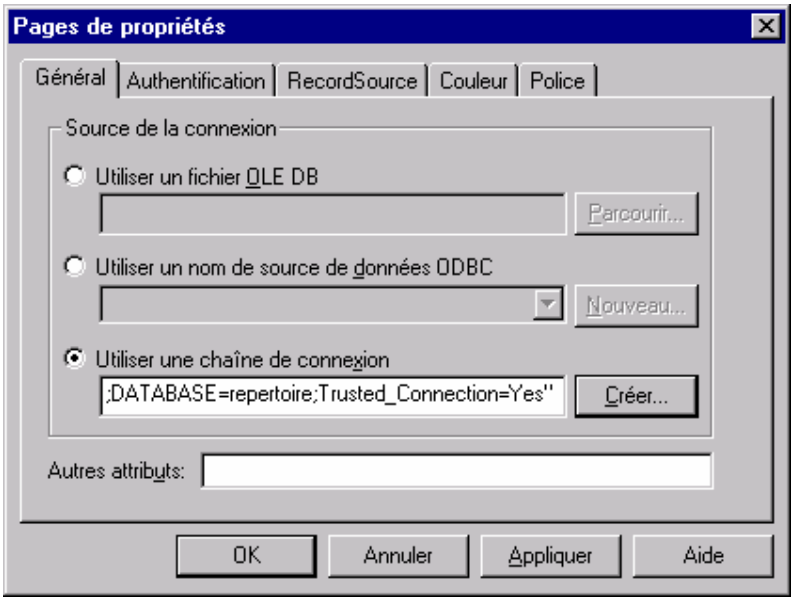
Table "**individu**" reprenant les champs suivant :

- id_individu	num(4,0)	clé primaire (autoincrement)
- nomprenom	char(30)	
- adresse	char (90)	not null
- telephone	char (15)	not null

- A partir du contrôle ADODC, créer une chaîne de connexion **OLE DB** permettant d'accéder à la Base De Données **SQL Server** "**repertoire**" se trouvant sue le serveur **PC093** :







Script de connexion :

Provider=MSDASQL.1;Persist Security Info=False;User ID=invité;Extended Properties="DRIVER=SQL Server;SERVER=pc093;APP= Visual Basic;WSID=PC093;DATABASE=repertoire;Trusted_Connection=Yes"

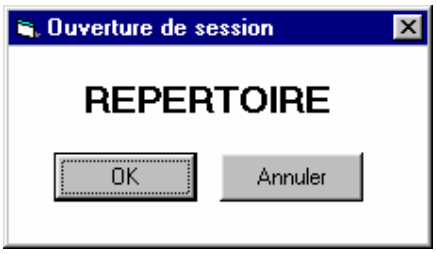
Projet :

Ajouter le composant Microsoft ADO Data Control 6.0 (SP3) OLE DB

Module :

Créer une instance d'objet de type global à partir de la classe ADODB.Connection

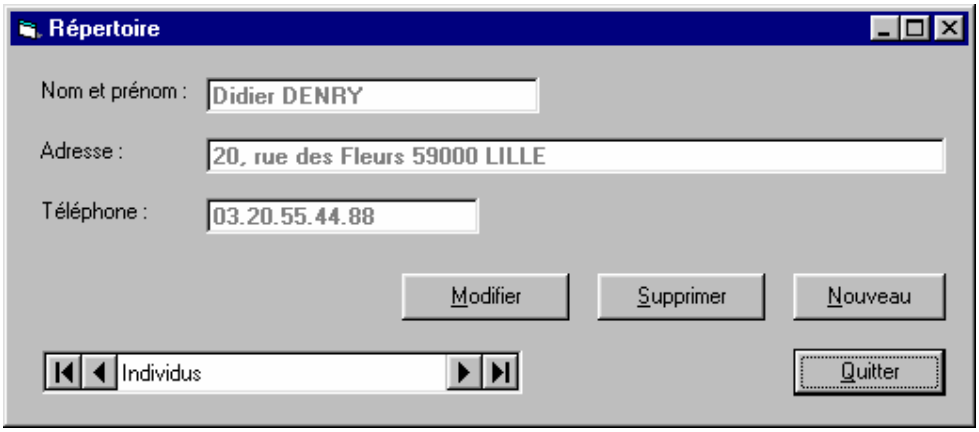
Feuille de connexion :



Sur Click du bouton OK, ouvrir la connexion de la BDD repertoire et afficher la feuille de consultation. En cas d'échec, envoyer un message d'erreur.

Feuille de consultation :

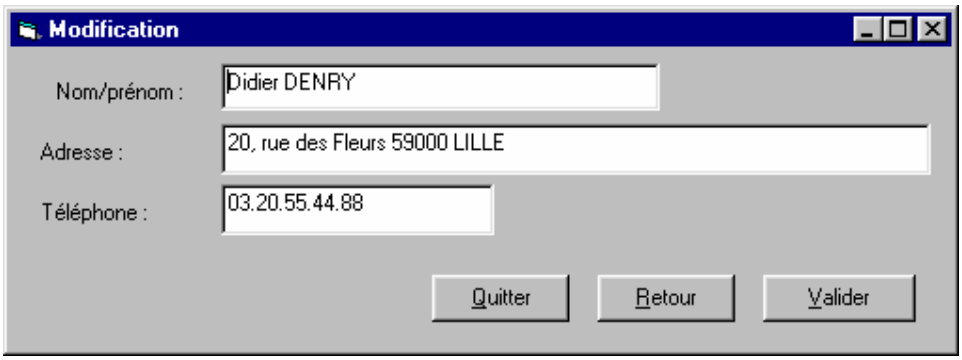
- des zones de texte pour la gestion des données du fichier Individu.
- un contrôle ADODC pour l'accès au fichier "INDIVIDU".
- un bouton Quitter pour fermer la Base de Données et quitter l'application.
- un bouton Supprimer pour supprimer l'enregistrement en cours.
- un bouton Modifier pour modifier l'enregistrement en cours.
- un bouton Nouveau pour ajouter un nouvel individu.



Sur le contrôle ADODC, préciser la chaîne de connexion et écrire la requête permettant de sélectionner tous les individus de la table individu dans l'ordre croissant des nom/prénom.

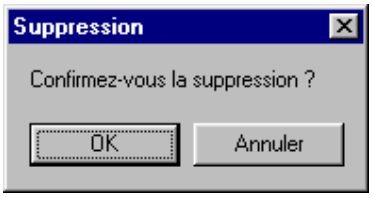
Feuille de modification :

Sur le bouton Valider, exécuter l'instruction SQL Update



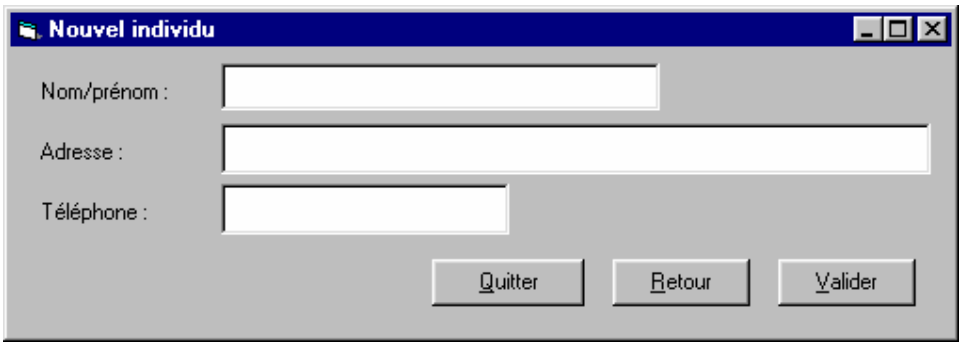
Feuille de Confirmation de suppression :

Sur le bouton Ok, exécuter l'instruction SQL Delete



Feuille de création :

Sur le bouton Valider, exécuter l'instruction SQL Insert



Vérifier la présence du nom sur les écrans de modification et d'ajout d'un nouvel individu :

Bases De Données
Eléments de correction

. Conception d'une BDD Relationnelle

. Construction d'une BDD

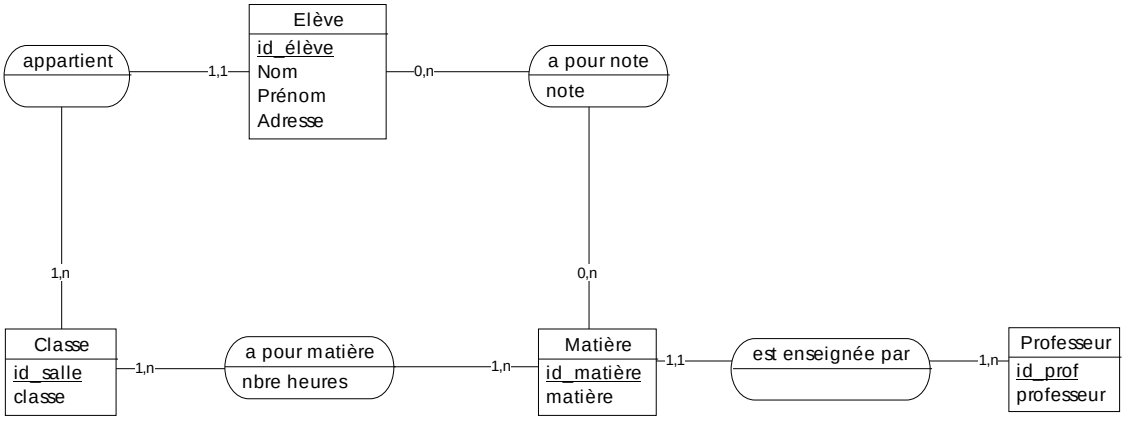
. Manipulation d'une BDD

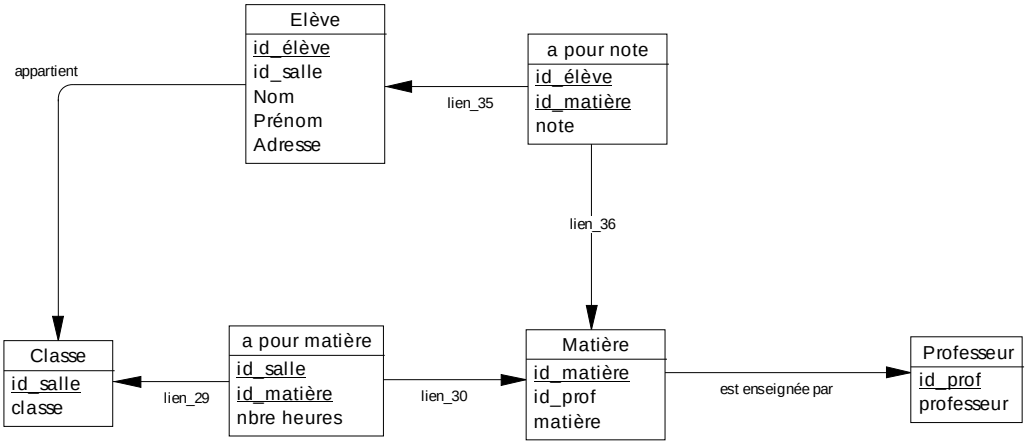
. Traitements dans les BDD

. Sécurités des BDD

. Accès aux BDD dans les langages hôtes

1 - Conception d'une BDD Relationnelle
Cas Note





```
%% ================================================================
%%   Nom de la base   :  CAS_NOTE
%%   Nom de SGBD      :  Sybase SQL Anywhere
%%   Date de création :  24/08/100  18:06
%% ================================================================

dbtool create database 'CAS_NOTE.db' transaction log to 'CAS_NOTE.log';
connect to 'CAS_NOTE.db' user "dba" identified by sql;

%% ================================================================
%%   Table : CLASSE
%% ================================================================
create table CLASSE
(
   ID_SALLE       char(10)            not null,
   CLASSE         char(10)            not null,
   primary key (ID_SALLE)
);

%% ================================================================
%%   Table : PROFESSEUR
%% ================================================================
create table PROFESSEUR
(
   ID_PROF        numeric(3)          not null,
   PROFESSEUR     char(40)            not null,
   primary key (ID_PROF)
);

%% ================================================================
%%   Table : ELEVE
%% ================================================================
create table ELEVE
(
   ID_ELEVE       numeric(6)          not null,
   ID_SALLE       char(10)            not null,
   NOM_ELEVE      char(20)            not null,
   PRENOM_ELEVE   char(20)            not null,
   ADRESSE_ELEVE  char(80)                  ,
   primary key (ID_ELEVE)
);

%% ================================================================
%%   Table : MATIERE
%% ================================================================
create table MATIERE
(
   ID_MATIERE     numeric(4)          not null,
   ID_PROF        numeric(3)          not null,
```

```
   MATIERE       char(40)          not null,
   primary key (ID_MATIERE)
);

%% ==========================================================
%%   Table : A_POUR_MATIERE
%% ==========================================================
create table A_POUR_MATIERE
(
   ID_SALLE      char(10)          not null,
   ID_MATIERE    numeric(4)        not null,
   NBRE_HEURES   numeric(2)              ,
   primary key (ID_SALLE, ID_MATIERE)
);

%% ==========================================================
%%   Table : A_POUR_NOTE
%% ==========================================================
create table A_POUR_NOTE
(
   ID_ELEVE      numeric(6)        not null,
   ID_MATIERE    numeric(4)        not null,
   NOTE          numeric(4,2)            ,
   primary key (ID_ELEVE, ID_MATIERE)
);

alter table ELEVE
   add foreign key FK_ELEVE_APPARTIEN_CLASSE (ID_SALLE)
      references CLASSE (ID_SALLE) on update restrict on delete restrict;

alter table MATIERE
   add foreign key FK_MATIERE_EST_ENSEI_PROFESSE (ID_PROF)
      references PROFESSEUR (ID_PROF) on update restrict on delete restrict;

alter table A_POUR_MATIERE
   add foreign key FK_A_POUR_M_LIEN_29_CLASSE (ID_SALLE)
      references CLASSE (ID_SALLE) on update restrict on delete restrict;

alter table A_POUR_MATIERE
   add foreign key FK_A_POUR_M_LIEN_30_MATIERE (ID_MATIERE)
      references MATIERE (ID_MATIERE) on update restrict on delete restrict;

alter table A_POUR_NOTE
   add foreign key FK_A_POUR_N_LIEN_35_ELEVE (ID_ELEVE)
      references ELEVE (ID_ELEVE) on update restrict on delete restrict;

alter table A_POUR_NOTE
   add foreign key FK_A_POUR_N_LIEN_36_MATIERE (ID_MATIERE)
      references MATIERE (ID_MATIERE) on update restrict on delete restrict;
```

1 - Conception d'une BDD Relationnelle
Cas Niche

1 - Conception d'une BDD Relationnelle
Cas FAP

1 - Conception d'une BDD Relationnelle
Cas Réservation Hôtel

3 - Manipulation d'une BDD
Langage de définition SQL
Base De Données Pubs

1 - Select nom_mag, ville from magasins where pays = 'FR' order by ville

2 - Select titre, nom_auteur, pn_auteur, pays
From titres a, titreauteur b, auteurs z
Where a.id_titre = b.id_titre and b.id_auteur = c.id_auteur and pays <> 'FR'
Order by pays, nom_auteur, pn_auteur

3 - Select nom_auteur, pn_auteur, pays, count(*)
From titreauteur a, auteurs b
Where a.id_auteur = b.id_auteur and pays <> 'FR'
Group by nom_auteur, pn_auteur, pays
Order by pays, nom_auteur, pn_auteur

4 - Select distinct a.pays, nom_auteur, pn_auteur, d.pays
From magasins a, ventes b, titreauteur c, auteurs d
Where a.id_mag = b.id_mag and b.id_titre = c.id_titre and c.id_auteur = d.id_auteur
and d.pays <> 'FR'
Order by nom_auteur, pn_auteur, a.pays

5 - Select nom_auteur, pn_auteur, sum(qt)
From auteurs a, titreauteur b, ventes c
Where a.id_auteur = b.id_auteur and b.id_titre = c.id_titre
Group by nom_auteur, pn_auteur
Order by 3 desc

6 - Select distinct nom_editeur, nom_auteur, pn_auteur
From editeurs a, titres b, titreauteur c, auteurs d
Where a.id_editeur = b.id_editeur and b.id_titre = c.id_titre and c.id_auteur =
d.id_auteur
Order by nom_auteur, pn_auteur, nom_editeur

7 - Select month(date_cmd) as mois, nom_mag, sum(qt)
From magasins a, ventes b
Where a.id_mag = b.id_mag
Group by mois, nom_mag
Order by 2, 1

3 - Manipulation d'une BDD
Langage de définition SQL
Base De Données Profor

```
% Exo 1
% Liste des noms de personnes des établissements 1 et 74

select persnom
        from tpers
        where uep_etab = '001' or uep_etab = '074'



% Exo 2
% Liste des numéros et noms des personnes de l'établissement 001
% entrés dans cet établissement entre 75 et 80

select pers, persnom, dateetab
        from tpers
       where uep_etab = '001' and
             dateetab between '1975/01/01' and '1980/12/31'


% Exo 3
% Liste des personnels ayant un T et un O dans leur nom

select persnom
         from tpers
         where persnom like '%T%' and persnom like '%O%'



% Exo 4
% Liste des personnels ayant deux E dans leur nom

select persnom
         from tpers
         where persnom like '%E%E%'



% Exo 5
% Liste des personnels ayant un code filière qui le même que le code UEP

select  persnom, pers, filiere_secteur, filiere_no, uep_secteur, uep_no
         from tpers
         where filiere_secteur = uep_secteur
                  and filiere_no = uep_no
                  and filiere_secteur is not null
```

```
% Exo 6
% Liste des noms des personnels, numéro d'établissement, salaire et numéro d'UEP
% triés par numéro d'établissement en croissant,
% numéro d'UEP en décroissant,
% salaire en décroissant
% et ceci uniquement quant le numéro d'UEP est fourni.

select distinct uep_etab, uep_secteur, uep_no, salmens, persnom
        from tpers
        where uep_secteur is not null
                and uep_no is not null
        order by uep_etab, uep_secteur desc, uep_no desc, salmens desc



% Exo 7
% Liste des personnels travaillant dans l'établissement de Saint-Avold
% Le code de Saint- Avold est supposé non connu.

select pers, persnom, perspren
        from tpers, tetab
        where uep_etab = etab
                and etabnom = 'Saint-Avold'
% Exo 7 bis
% Liste des personnels travaillant en Seine et Marne



select pers, persnom, perspren
        from tpers, tetab, tdept
        where uep_etab = etab
                and tetab.dept = tdept.dept
                and deptnom = 'Seine et Marne'



% Exo 8
% Liste des personnes rattachées à une UEP (GTS)  avec le nom de cette UEP
% et le nom du secteur de rattachement.

select persnom, perspren, filierenom, sectnom
        from tpers p, tgts u, tsecteur
        where p.filiere_secteur = u.filiere_secteur
                and p.filiere_no = u.filiere_no
                and u.filiere_secteur = secteur
```

```
% Exo 9
% Liste des noms de personne qui sont entrés dans l'organisme avant Dominique
PESSON

select  a.persnom, a.dateent, b.dateent
        from tpers a, tpers b
        where b.persnom = 'PESSON'
                and a.dateent < b.dateent



% Exo 10
% Liste des noms de personne qui sont entrées la même année que Dominique PESSON

select a.persnom, a.dateent
        from tpers a, tpers b
        where b.persnom = 'PESSON' and b.perspren = 'Dominique'
                and year(a.dateent) = year(b.dateent)
                and a.pers <> b.pers



% Exo 11
% Liste des noms de spécialités qui ont le même niveau de quallification que OVNI

select spenom, nivqualif
        from tspecia
        where nivqualif in
                (select nivqualif from tspecia
                        where spenom = 'OVNI')
        and spenom <> 'OVNI'


% Exo 12
% Liste des personnes qui sont entrées avant toutes celles qui sont dans
% l'établissement 001

select persnom,pers, dateent
        from tpers
        where dateent <  all
                (select dateent  from tpers
                        where uep_etab = '001')
```

```
% Exo 13
% Liste des personnes ayant le même niveau dans le même établissement que
% Jacques DEODAT

select persnom,pers, niveau, uep_etab
        from tpers
        where persnom <> 'DEODAT' and niveau in
                (select niveau from tpers where persnom = 'DEODAT')
        and uep_etab in
                (select uep_etab from tpers where persnom = 'DEODAT')



% Exo 13 bis
% Liste des personnes ayant le même niveau dans un même établissement

select a.persnom, a.pers, nivnom, etabnom
        from tpers a, tpers b, tstatut c, tetab d
        where a.niveau = b.niveau and a.pers <> b.pers
        and a.uep_etab=etab and a.niveau=c.niveau
        and a.uep_etab = b.uep_etab



% Exo 14
% Liste des noms des responsables d'établissement dans lesquels
%certaines uep n'ont pas de responsable, en indiquant les uep

select persnom,  etabnom, a.uep_secteur, a.uep_no
        from tpers  , tetab , tuep a
        where respuep is null
        and a.uep_etab = etab
        and pers = respetab



% Exo 15
% Liste des noms des responsables d'établissement dans lesquels
%certaines uep n'ont pas de responsable, en  indiquant le nombre d'uep
% par établissement

select persnom,  etabnom, nbuepvide = count(a.uep_secteur+a.uep_no)
        from tpers  , tetab , tuep a
        where respuep is null
        and a.uep_etab = etab
        and pers = respetab
        group by persnom, etabnom
```

```
% Exo 16
% Liste des noms des personnes avec leur salaire arrondi au millier de francs inférieur
% que l'on nommera salmil

select persnom,  salmil=ceiling("truncate"(salmens,0))
        from tpers



% Exo 17
% Liste des salaires des personnels en masquant ceux de l'établissement 112
% avec des étoiles

select persnom, salaire = ('***************')
        from tpers
        where uep_etab='112'
union
select persnom, salaire=string(salmens)
        from tpers
        where uep_etab<>'112'



% Exo 18
% Liste des codes et noms de département qui ont plus d'un établissement rattaché

select dept, deptnom
        from tdept
        where dept in
                (select dept from tetab
                        group by dept
                        having count(etab)> 1)



% Exo 19
% Liste des numéros de programmes des actions élémentaires avec les dates
%de début de mois de l'action, classées par dates de début d'action

select  mois = date(dateact - day(dateact)+1), prog
        from tact
        where dateact is not null
        order by mois
```

```
% Exo 20
% Liste des personnels( numéro nom état civil nombre d'enfants) avec le nombre
%de mois depuis leur entrée dans l'établissement, classée par établissement
%et par ancièNneté dans l'établissement

select  pers, persnom, etatcivil, nbenf, uep_etab,
        nbmois = (months(now(*)) - months(dateetab))
       from tpers
       order by  uep_etab, nbmois




% Exo 21
%Donner par établissement et par uep d'un établissement, les salaires annuels
%moyens en 1991. L e nombre de mois de paye dans l'année de référence
%est dans la table des données générales

select  uep_etab, uep_secteur, uep_no, moysal =cast(avg(salmens*nbmois) as integer)
        from tpers, tdg
       where an = 1991
       group by  uep_etab, uep_secteur, uep_no
       order by 1, 2, 3




% Exo 22
%Liste des n° et noms de personnes qui sont entrées dans l'organisme avant d'entrer
%dans l'établissement en indiquant leur ancienneté en mois à ce moment là

select  pers, persnom, perspren,
          anciennete = months(dateetab) - months(dateent)
        from tpers
       where dateetab > dateent



% Exo 23
%Tout le personnel de niveau 40 va voir son salaire augmenter de 10%.
%remettre la base de données en état
commit;
update tpers set salmens = salmens*1.1 where niveau = 40;
select persnom, perspren, salmens from tpers where niveau = 40;
rollback
```

```
% Exo 24
%Créer une vue avec nom de région, référence, et noms des spécialités
%pratiquées dans cette région
%utiliser cette vue pour obtenir le nombre de spécialités par région,
%puis la liste de ces noms de spécialités par régions.
%ensuite détruire la vue

create view regspe as
      select specia, spenom, regnom
            from tspecia, tgts, tuep, tetab, tdept, treg
            where tspecia.filiere_secteur = tgts.filiere_secteur
                  and tspecia.filiere_no = tgts.filiere_no
                  and  tuep.uep_secteur = tgts.filiere_secteur
                  and tuep.uep_no = tgts.filiere_no
                  and tuep.uep_etab = tetab.etab
                  and tetab.dept = tdept.dept
                  and tdept.reg = treg.reg;

select regnom, nbspe = count(*)
      from regspe
      group by regnom;
drop view regspe;


% Exo 25
%donner par établissement et par uep, les salaires annuels moyens de 1990.
%A chaque établissement, associer son nom et à chaque uep, le nom de la
%filière correspondante.

drop view moyetab;
drop view moyuep;
create view moyetab(etab, smae) as
      select uep_etab, moysaletab = cast(round(avg(salmens*nbmois),3) as integer)
            from tpers, tdg
            where an = 1990
            group by uep_etab;

create view moyuep(etab, uep_secteur, uep_no, smau) as
      select  uep_etab, uep_secteur, uep_no,
                  moysaluep = cast(round(avg(salmens*nbmois),3) as integer)
            from tpers, tdg
            where an = 1990
            group by uep_etab, uep_secteur, uep_no;

select tetab.etab, tetab.etabnom, tgts.filierenom, moyuep.smau, moyetab.smae
      from tetab, moyetab,    moyuep left outer join tgts on
      ( moyuep.uep_secteur = tgts.filiere_secteur and
```

```
            moyuep.uep_no = tgts.filiere_no)
      where    moyetab.etab = tetab.etab
            and moyuep.etab = tetab.etab
       order by tetab.etab;


% Exo 26
%Créer la procédure stockée lstetab qui renvoie les n°, nom
% et prénom des personnes d'un établissement donné.
% Le n° établissement est le paramètre.
%Les colonnes retournées s'appelleront ppersno, ppersnom, pperspren.
% Essayer la procédure.

create procedure lstetab
       (in etabno char(3))
      result (  " ppersno" char(5), "ppersnom" char(30), "pperpren" char(30) )
begin
      select pers, persnom, perspren
            from tpers, tetab
            where etab = etabno
         and etab = uep_etab;
end


% Exo 27
% Créer une fonction permettant de concaténer le prénom et le nom d'une personne
% en supprimant les espaces inutiles
%Essayer cette fonction sur les personnes de Saint Avold

create function fullname (perspren char(30), persnom char(30))
returns char(61)
begin
      declare nom char(61);
      set nom = perspren || ' ' || persnom;
      return(nom);
end


% Exo 28
% Créer une fonction permettant de concaténer le secteur et le no d'une filière

create function filiere (sec char(2), no char(2))
returns char(4)
begin
      declare nfiliere char(4);
      set  nfiliere  = sec + no;
      return(nfiliere);
end
```

```
% Exo 29
% Lister les personnes (nom et prénom) en indiquant le nom deleur filière.
% Utiliser les fonctions des exos 27 et 28

select personne = fullname(perspren, persnom), filiere = filierenom
        from tpers, tgts
        where  filiere(tpers.filiere_secteur, tpers.filiere_no) =  filiere(tgts.filiere_secteur,
tgts.filiere_no)



%exo30
%Créer un trigger permettant de contrôler que la date d'établissement est supérieure
%ou égale à la date d'entrée lors de la mise à jour d'une personne (insert, update)

create trigger verif_entre_majpers
      before update, insert on tpers
      referencing new as new_tpers for each row
begin
      declare err_1 exception for sqlstate '99999';

      if new_tpers.dateetab < new_tpers.dateent  then
            signal err_1;
      end if;

exception
when err_1 then
      message 'Erreur: Trigger(verif_entre_majpers) de la table TPERS';
      message '       La date établissement inférieure à date entrée !';
      signal err_1;
when others then
      message 'Exception dans trigger(verif_entre_majpers) avant insertion de la table
TPERS';
      resignal;
end
```

6 - Accès aux BDD dans les langa ges hôtes

Module Module1

```
Global cnn As New ADODB.Connection
```

Page de connexion frmLogin

```
Option Explicit

Sub ouvertureBdd()
   On Error GoTo ErreurConnexion
   cnn.Open "Provider=MSDASQL.1;Persist Security Info=False;User
ID=pc093\invité;Extended Properties='DRIVER=SQL
Server;SERVER=pc093;APP=Visual
Basic;WSID=PC093;DATABASE=repertoire;Trusted_Connection=Yes'"
   Exit Sub

ErreurConnexion:
   MsgBox "Erreur " & Err & ": " & Error
End Sub

Private Sub cmdCancel_Click()
   End
End Sub

Private Sub cmdOK_Click()
   MousePointer = vbArrowHourglass
   ouvertureBdd
   Me.Hide
   FrmIndividu.Show
End Sub
```

Page de consultation frmIndividu

```
Dim book As Variant

Private Sub CmdModifier_Click()
   With Adodc1.Recordset
      vMyBookMark = .Bookmark
   End With
   FrmModifier.Show
End Sub

Private Sub CmdNouveau_Click()
   FrmNouveau.Show
End Sub
```

```
Private Sub CmdQuitter_Click()
   End
End Sub

Private Sub CmdSupprimer_Click()
   Dim Réponse As Integer
   Dim strDelete As String
   On Error GoTo ErreurDelete
   Réponse = MsgBox("Confirmez-vous la suppression ?", vbOKCancel, "Suppression")
   If Réponse = vbOK Then
      cnn.BeginTrans
         With cnn
            strDelete = "delete from INDIVIDU where id_individu = " &
                  Adodc1.Recordset.Fields(0)
            .Execute strDelete
         End With
      cnn.CommitTrans
      MousePointer = vbArrowHourglass
      Adodc1.Refresh
      MousePointer = vbDefault
      FrmIndividu.Caption = "Répertoire de " & Adodc1.Recordset.RecordCount &
          " individus"
   End If
   Exit Sub

ErreurDelete:
   cnn.RollbackTrans
   MsgBox "Incident lors du Delete : " & Err & " " & Error

End Sub

Private Sub Form_Activate()
   MousePointer = vbArrowHourglass
   With Adodc1
      .Refresh
      FrmIndividu.Caption = "Répertoire de " & .Recordset.RecordCount & " individus"
      If vMyBookMark <> 0 Then
         .Recordset.Bookmark = vMyBookMark
      End If
   End With
   MousePointer = vbDefault
End Sub

Private Sub Form_Load()
   MousePointer = vbDefault
End Sub

Private Sub Form_Terminate()
   cnn.Close
```

```
End Sub

Public Property Get id() As Double
    id = Adodc1.Recordset.Fields(0)
End Property


Public Property Get vMyBookMark() As Variant
   vMyBookMark = book
End Property

Public Property Let vMyBookMark(ByVal vNewValue As Variant)
   book = vNewValue
End Property
```

Page de création frmNouveau

```
Private Sub CmdQuitter_Click()
   End
End Sub

Private Sub CmdRetour_Click()
   Unload Me
End Sub

Private Sub CmdValider_Click()
   Dim strInsert As String
   If txtNomPrénom = "" Then
      MsgBox "Nom obligatoire"
      txtNomPrénom.SetFocus
      Exit Sub
   End If
   On Error GoTo ErreurInsert
   With cnn
      .BeginTrans
      strInsert = "insert into INDIVIDU values('" & txtNomPrénom.Text & "', '"
      strInsert = strInsert & txtAdresse.Text & "', '" & TxtTéléphone.Text & "')"
      .Execute strInsert
      .CommitTrans
   End With
   txtNomPrénom = ""
   txtAdresse = ""
   TxtTéléphone = ""
   Exit Sub

ErreurInsert:
   cnn.RollbackTrans
   MsgBox "Incident lors de l'Insert" & Err & ": " & Error
```

```
End Sub
```

Page de modification frmModifier

```
Private Sub CmdQuitter_Click()
   End
End Sub

Private Sub CmdRetour_Click()
   Unload Me
End Sub

Private Sub CmdValider_Click()
   Dim strUpdate As String
   If TxtNomPrénom.Text = "" Then
      MsgBox "Nom obligatoire"
      TxtNomPrénom.SetFocus
      Exit Sub
   End If
   On Error GoTo ErreurUpdate
   With cnn
      .BeginTrans
      strUpdate = "update INDIVIDU "
      strUpdate = strUpdate & "set nomprenom = '" & TxtNomPrénom.Text & "', "
      strUpdate = strUpdate & "adresse = '" & TxtAdresse.Text & "', "
      strUpdate = strUpdate & "telephone = '" & TxtTéléphone.Text & "' "
      strUpdate = strUpdate & "where id_individu = " & FrmIndividu.id
      .Execute strUpdate
      .CommitTrans
   End With
   Unload Me
   Exit Sub

ErreurUpdate:
   cnn.RollbackTrans
   MsgBox "Incident lors de l'Update" & Err & ": " & Error
End Sub

Private Sub Form_Load()
   Dim strSelect As String
   Set rs = New ADODB.Recordset
   With rs
      strSelect = "select * from individu where id_individu = " & FrmIndividu.id
      .Open strSelect, cnn
      TxtNomPrénom.Text = .Fields(1)
      TxtAdresse.Text = .Fields(2)
      TxtTéléphone.Text = .Fields(3)
   End With
End Sub
```

Généralités SGBD :

Systèmes de Gestion de Base de Données

SYSTEME DE GESTION DE BASE DE DONNEE

Les bases de données : PRESENTATION

Pourquoi une base de données ?

Une entreprise doit conserver la trace d'un volume élevé d'informations. Ces dernières peuvent être, par exemple, les noms et salaires des employés, les adresses des fournisseurs, le montant des stocks ou le montant des produits de l'exercice.

Traditionnellement, différentes parties de ces informations sont conservées par différents départements et/ou différents individus, chacun ayant la charge des données qu'il utilise le plus souvent. Pour obtenir une information, il faut d'abord déterminer où elle est stockée, puis s'adresser au département ou à la personne concernée.

Les comptables utilisent très fréquemment un grand nombre de données concernant l'entreprise. Quand il prépare les bilans ou un relevé financier, le comptable dispose habituellement de la plupart des informations nécessaires, qui sont le plus souvent contenues dans les grands livres ou les journaux. Cependant, certaines données supplémentaires (en particulier lors de la préparation des budgets), telles que les coûts standards et unitaires, peuvent ne pas apparaître dans ces documents financiers et comptables. Il faut consulter d'autres départements pour obtenir des précisions ou des renseignements supplémentaires.

Les entreprises qui ont informatisé une grande part de leurs fonctions de gestion ont souvent de nombreux programmes d'application différents, chacun ayant un but différent, comme la préparation des bulletins de salaire, la gestion des stocks, la facturation etc. Chacun de ces programmes utilisera vraisemblablement ses propres fichiers contenant les données nécessaires.

Les systèmes manuels ou informatisés en partie ou en totalité que nous venons d'évoquer ont plusieurs défauts majeurs. Dans tous ces systèmes, il peut être nécessaire de s adresser à plusieurs départements avant de trouver celui qui peut effectivement fournir les données. D'autre part, les informations utilisées par plusieurs départements peuvent être conservées en plusieurs endroits, ce qui entraîne une redondance de stockage, Ceci entraîne un gaspillage au niveau du volume des fichiers et des coûts supplémentaires souvent élevés.

En outre, s'il y a redondance dans les données, toute modification doit être faite plusieurs fois, afin de maintenir la cohérence entre les différents fichiers. S'il se produit une erreur, des incohérences peuvent apparaître. Par exemple, le directeur de la production et le directeur des ventes peuvent conserver tous les deux une trace du stock courant d'un article. Au moment de la vente, les deux enregistrements doivent être mis à jour, pour refléter la baisse du stock. Si un des enregistrements indique un stock inférieur ou supérieur à la réalité, il peut en résulter une sous ou surproduction, ou une vente peut être perdue, avec chaque fois un coût supplémentaire pour l'entreprise.

D'autres soucis concernent la sécurité des données et les accès non autorisés à des informations confidentielles. Accéder à l'information est simple s'il suffit pour cela d'ouvrir un tiroir de bureau; cependant, les données stockées dans une application informatisée sont disponibles chaque fois que cette application est utilisée.

Les programmes d'application tendent aussi à utiliser des méthodes de stockage des données assez complexes et l'utilisateur doit souvent se préoccuper de savoir quel format utiliser pour ses données.

Pour ces raisons une tendance s'est développée pour combiner toutes les informations importantes de l'entreprise dans une base de données intégrée. Dans une base de données, le stockage des données est entièrement centralisé. Dans l'idéal, il n'existe qu'un exemplaire de chaque élément de données. Les mises à jour ne sont donc exécutées qu'une seul fois et les problèmes d'incohérence sont limités.

Chaque utilisateur peut ensuite avoir accès aux données spécifiques dont il a besoin. On inclut des mesures de protection pour éviter que quelqu'un puisse accéder à des informations qu'il n'est pas autorisé à connaître (par exemple, les salaires des employés), et pour rendre immédiatement disponibles les données voulues. Chaque département dispose d'un ou plusieurs programmes d'application, pour lire les données, traiter les transactions et effectuer les mises à jour. Les complexités du stockage ne sont pas apparentes pour l'utilisateur, qui n'a connaissance que des données dont il a besoin.

Qu'est-ce qu'une base de données?

Le premier problème auquel on se trouve confronté est de déterminer précisément ce que recouvrent les termes "base de données" et "système de gestion de bases de données" (SGBD). Un usage non-averti de ces termes peut se référer en fait à toute une collection de données accessible par l'intermédiaire d'un ordinateur.
Une base de données a trois caractéristiques essentielles. C'est d'abord un ensemble organisé et intégré de données. Elle correspond ensuite à une représentation fidèle des données et de leur structure, avec le minimum possible de contraintes imposées par le matériel. On doit enfin pouvoir l'utiliser pour toutes les applications pratiques désirées sans duplication de données.
Il existe trois types de bases de données : hiérarchiques, réseaux, relationnelles. Les caractéristiques liées à chacun de ces types sont développées plus loin.
Le **S**ystème de **G**estion de **B**ase de **D**onnées est le logiciel qui supporte une telle organisation des données. On peut le définir plus précisément comme, un ensemble de logiciels fournissant l'environnement pour décrire, mémoriser, manipuler et traiter des ensembles de données tout en assurant pour celle-ci la sécurité, la confidentialité et l'intégrité (la notion d'intégrité est proche de celle d'exactitude de cohérence), sachant qu'un grand nombre d'utilisateurs ayant des besoins variés interagit avec ces ensembles de données.

Les définitions sont cependant de peu d'intérêt pour déterminer si un système est vraiment un SGBD ou s'il s'agit simplement d'un système d'information classique ou d'un système de fichiers. Il faut mieux définir le SGBD en précisant certaines des fonctions qu'il doit remplir :

— l'intégration des données afin d'éviter l'incohérence d'éventuelles données dupliquées (tout est intégré dans un seul ensemble cohérent) ;

— la séparation entre les moyens de stockage physique des données et la logique des applications ;

— un contrôle unique de toutes les données afin de permettre l'utilisation simultanée par plusieurs utilisateurs ;

— la possibilité d'utiliser des structures de fichiers et des méthodes d'accès complexes, de façon à ce que les relations correctes entre les données puissent être exprimées et les données utilisées le plus efficacement dans un grand nombre d'applications ;

— des facilités pour le stockage, la modification, le réorganisation, l'analyse et la consultation des données, sans que le système impose des restrictions à l'utilisateur ;

— des contrôles de sécurité afin d'empêcher l'accès illégal à certaines données ;

— des contrôles d'intégrité pour prévenir une modification indue des données (exemple: contrôle d'exactitude, de validité) ;

— la compatibilité avec les principaux langages de programmation, les programmes-sources existants, et les données extérieures à la base.

Les SGBD les plus évolués actuellement disponibles disposent de la plupart de ces fonctions, mais pas toutes. Il en résulte des différences significatives entre leurs caractéristiques, leur fonctionnement et leurs usages possibles.

Il convient de signaler également que les bases de données sont plus qu'une nouvelle technique de stockage et de manipulation des données. Elles impliquent une nouvelle approche de la conception et de l'utilisation des systèmes d'information et peuvent avoir des conséquences organisationnelles qui sortent largement du cadre du service informatique. Elles obligent les utilisateurs à considérer les données comme une ressource de l'entreprise qui doit être gérée comme le sont les ressources traditionnelles (personnel, locaux, moyens de production, capitaux) pour être accessible à un grand nombre d'utilisateurs.

Différences entre les bases de données et les systèmes traditionnels de gestion des fichiers

Les caractéristiques principales d'une base de données sont :

— l'indépendance de la structure des données par rapport aux programmes de traitement (cette structure de données est prévue une fois pour toutes, pour tous les programmes : une base de données peut et doit évoluer mais doit vivre des périodes stables assez longues de 6 mois à un an) ;

— la prise en compte des relations entre les différentes données (chaînages) ;

— la non-redondance des données (en principe !) ;

— le partage simultané des données entre les programmes d'applications ;

— l'entité élémentaire que l'on peut lire dans une base de données est beaucoup plus petite qu'un enregistrement de fichier : elle correspond à un segment ou record.

Compte tenu de ces caractéristiques, il est évident que l'approche des systèmes traditionnels de fichiers diffère considérablement du concept de base de données

Avantages et inconvénients des SGBD

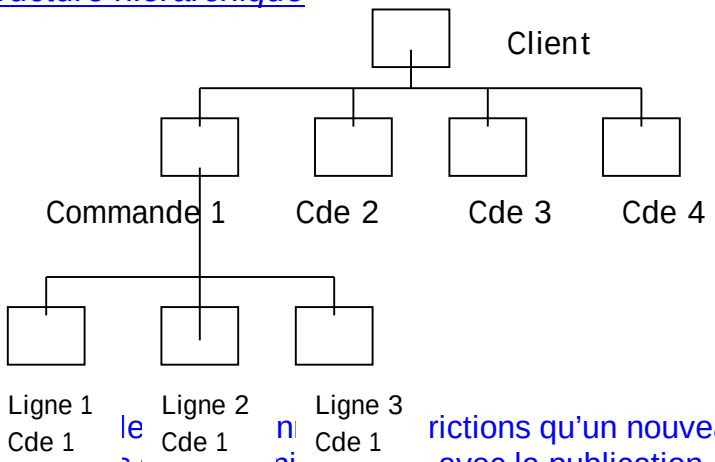
Les gestionnaires de base de données de type hiérarchique sont apparus à la fin des années 1960 (comme DL/1, IMS ou Système 2000).

Ces gestionnaires permettent de traiter de façon élégante et efficace une situation très fréquemment rencontrée dans la pratique : la dépendance Mono-dimensionnelle. Un exemple type est le problème CLIENT-COMMANDE : chaque client est "propriétaire" d'un certain nombres de commandes qu'il a passées.

Pourtant, si on ajoute un étage à cette construction, on atteint les limites du hiérarchique.
En effet, chaque commande se décompose en "ligne de commande" qui référencent un produit et une quantité commandée pour ce produit. Le schéma hiérarchique ne permet d'établir un lien fonctionnel entre la ligne commande et le produit commandé qu'à travers une redondance de donnée (le code "produit" figure dans la liste des items de l'enregistrement LIGNE).

On retombe dans la même ornière qu'avec un gestionnaire de fichiers.
Pourquoi ? Essentiellement parce que le hiérarchique ne peut traiter que des situations de décomposition mono-dimensionnelle alors que nous sommes en présence d'un problème à deux dimensions : la quantité dépend à la fois de la commande dans laquelle elle existe et aussi du produit qui est référencé. Or, un gestionnaire hiérarchique interdit à un enregistrement d'avoir plusieurs parents : c'est une limitation fonctionnelle, dont nous voyons ici les conséquences pratiques

Structure hiérarchique



C le n rictions qu'un nouveau type de gestionnaire de base de données a été donné en 1971, avec la publication, par un groupe de travail réunissant fabricants d'ordinateurs et utilisateurs, du rapport connu sous le nom de "**Codasyl** Data Base Task Group Report".

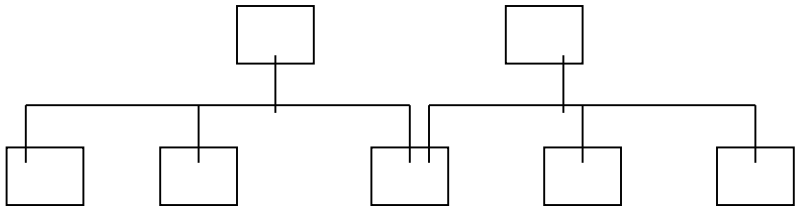
Ce rapport reprenait, en les développant, des idées qui venaient d'être mises en œuvre par Charles Bachman dans la conception du SGBD IDS sur le matériel Honeywell. Ce rapport eut un retentissement suffisant pour que, peu après, deviennent disponibles des SGBD de types **réseau**, "aux normes Codasyl" aussi célèbres maintenant que: IDMS (de Cullinane), IDS II (sur Honeywell), DMS 1100 (Univac), DSMS (Data General) et DBMS (Prime).

Dans cette nouvelle génération de SGBD, on parle désormais de : type d'enregistrement et de relation (set) ; toute relation a (au moins) un propriétaire et un membre, qui sont des types d'enregistrement ; et les types d'enregistrement peuvent être à la fois membre et propriétaire de relations, sans limitation fonctionnelle.

Il faut noter que, dans les deux modèles de base déjà étudiés (hiérarchique et réseaux), la souplesse apparente due à la possibilité de faire créer par le système

tous les pointeurs désirés est en fait limitée par la nécessité de définir, au moment de la création de la base, l'ensemble des pointeurs et chemins d'accès voulus. Toute modification ultérieure impliquant généralement une refonte de la base et des programmes d'exploitation.

Structure réseau



D'autre part, il existe, en plus des problèmes techniques, une "composante humaine" fort importante. En effet, si la compréhension d'un système de gestion de base de données (SGBD), est à la portée de tous ceux qui ont une culture informatique normale, il n'en demeure pas moins que la mise en œuvre de ce système est complexe, laborieuse et, à tout le moins, riche en embûches de toutes sortes et en problèmes de dernière minute, même si l'analyse a été bien menée.

On peut distinguer deux types de difficultés : d'une part, celles qui ont trait à la conception et à la mise sur pied de la base ; d'autre part, celles qui sont liées à son utilisation et à son évolution.

Dans le premier cas, l'utilisateur est confronté au double problème de la définition de son application et de la compréhension du SGBD choisi.
Il suffit de voir la taille de la documentation technique fournie avec certains produits, IMS d'IBM par exemple, pour se rendre compte de la complexité d'un logiciel de base de données.
L'utilisateur va devoir assimiler le langage de commande de son SGBD pour pouvoir définir sa base et les chemins d'accès. Pas toujours simple !

Dans le second cas, il va s'agir des ennuis habituels rencontrés lors de la mise en route d'une application informatique, quelle qu'elle soit. Toutefois, ils vont revêtir souvent un caractère de gravité car, contrairement aux applications classiques, celles qui utilisent des bases de données seront généralement dans un contexte "temps réel" avec de nombreux utilisateurs en ligne.
Dans ce cas la mise au point est délicate, car toute modification de la structure de la base entraîne automatiquement un "arrêt" temporaire de l'activité de la base en question.
Il est, en effet, impensable qu'un utilisateur puisse travailler sur une base que l'on est train de transformer.

Une des raisons pour lesquelles ces problèmes existent est sans aucun doute l'existence de liens physiques qui doivent être spécifiés au moment de la construction de la base et dont la modification ultérieure n'est pas permise par le SGBD sans reconstruction de la base.
C'est en effet le système qui va constituer la base de données en stockant les informations que l'on va lui fournir dans un ordre lié à la structure qui aura été définie. Les index vont être construits et stockés également d'une façon liée à l'organisation hiérarchique.

On a donc imaginé une autre organisation de base de données dans laquelle ces contraintes n'existeraient plus. Pour cela, une notion nouvelle doit se faire jour, celle de base de données "**relationnelle**". Terme à la mode, sans doute un peu ésotérique pour beaucoup, mais qui finalement va se comprendre sans trop de difficultés pour la bonne raison qu'il est beaucoup plus proche du raisonnement humain, que le terme hiérarchique ou arborescent.

Prenons donc un exempte calqué sur le raisonnement humain. Un adulte normalement cultivé possède "en mémoire" des millions d'informations rangées dans le cerveau, selon des mécanismes qui, pour la plupart, nous échappent encore totalement.
Ces informations peuvent être de nature différentes : mots, sensations, odeurs, bruits et sons, notions abstraites, etc.
Lorsque l'on nous pose une question ou que la discussion implique une recherche, il va de soi que nous n'allons pas consulter la totalité de notre mémoire, ni même toutes les informations que nous possédons sur le sujet donné, avant de donner notre réponse.
Premièrement, nous en serions bien incapables et ensuite il nous faudrait un temps rédhibitoire.
La raison en est fort simple : le cerveau humain n'est absolument pas construit selon les concepts de l'informatique actuelle. Sa structure n'est pas celle d'un disque magnétique et sa mémoire n'est pas "adressable". Chose curieuse d'ailleurs puisqu'il est quand même possible de retrouver instantanément une information ! Pour l'homme, c'est le contenu de la question qui va servir pour déduire la réponse. Il y a un processus déductif qui est à l'opposé du processus inductif utilisé dans la recherche sur une base de données hiérarchisée.
Lorsqu'on nous demande quelle est notre profession, il est évident que nous n'avons pas besoin de passer mentalement en revue tous les métiers que nous connaissons pour trouver le nôtre. Nous le trouvons immédiatement parce que la question est posée de telle façon que les mots "profession" et "nôtre" réduisent le champ de recherche à son strict minimum.
Notons encore qu'il est possible de poser la même question sous une bonne centaine de formes différentes quant à l'ordre des mots, leur nombre, ou bien la langue employée.

Les bases de données relationnelles fonctionnent donc selon ce principe : c'est le contenu de la question qui va déterminer quels sont les chemins d'accès à utiliser, les liens à établir.
Ces liens, ou pointeurs, ne seront plus fixés une fois pour toutes dans le SGBD mais ils seront fabriqués dynamiquement selon les besoins.
Mais cela n'est pas suffisant car il faudra que le langage d'interrogation du système soit capable de comprendre correctement une question. Pour cela, une seule possibilité : l'utilisation des opérateurs logiques de la théorie des ensembles (autre formulation l'algèbre relationnelle).
Le SGBD agira "par déduction" à partir de l'analyse de la question. Telle question implique la consultation des fichiers W et X, telle autre implique la consultation des fichiers Y et Z. Bien entendu, il y aura des liens entre question et fichiers mais ils existeront non comme pointeurs mais comme éléments d'un tableau géré par le SGBD. Que trouverons-nous dans ce tableau ? Tout simplement les associations entre les divers "champs" ou données élémentaires, et les fichiers correspondants.
Il sera possible à tout moment d'ajouter ou de supprimer des fichiers dans la base de données. Les associations seront faites lors des interrogations de la base, en fonction des noms de données spécifiés.

Par exemple, la donnée appelée "MATRICULE" se trouve dans les fichiers "PAIE", "HISTORIQUE" et "PERSONNEL". Dans ce cas, toute question faisant intervenir le mot "MATRICULE" quel que soit le contexte, déclenchera le chaînage logique des fichiers désignés. Grâce à son tableau de relations le SGBD va associer tous les fichiers contenant un ou plusieurs champs spécifiés dans la question de l'utilisateur.

Il est évident qu'une base de données relationnelle présente des avantages tels que :

- L'indépendance des utilisateurs vis à vis de la structure logique, la structure physique, la stratégie d'accès aux données;

- La puissance de représentation grâce à conception rigoureuse du schéma (approche méthodologique);

- La rapidité d'écriture du code

- La manipulation par des non informaticiens pour des cas simples sinon la connaissance de la structure de la base de données et de la maîtrise de l'algèbre relationnelle sont nécessaires.

- L'approche non procédurale permettant un traitement uniforme de la définition, de la manipulation et du contrôle des données

- L'évolutivité

- La prise en compte des contraintes d'intégrité

SOMMAIRE

LES DIFFERENTS SGBD

L'OUTIL BASE DE DONNEES : GENERALITES

1) Définition

Le concept Base de Données (BDD) apparaît vers 1960 suite au nombre croissant d'informations que les entreprises doivent gérer et partager (évolution dans les systèmes informatiques). Historiquement, chaque nouvelle application engendrait ses propres fichiers et ses propres programmes. La création d'une base de données va à l'encontre de cette façon de procéder ; elle rend possible la centralisation, la coordination, l'intégration et la diffusion de l'information archivée.

On peut en donner la définition suivante : une base de données est un ensemble structuré de données enregistré sur des supports accessibles par l'ordinateur, pour satisfaire simultanément plusieurs utilisateurs de façon sélective et en un temps opportun.

En fait, une base de données enregistre des faits, des événements qui surviennent dans la vie d'un organisme, pour les restituer à la demande ; elle permet également de tirer des conclusions en rapprochant plusieurs faits élémentaires les uns des autres. Les données peuvent être manipulées par plusieurs utilisateurs ayant des vues différentes de ces données. La structure de l'ensemble requiert une description rigoureuse appelée **SCHEMA**.

2) Facteurs à l'origine du développement

- nombre croissant d'informations à gérer.
- évolution technologique des mémoires : augmentation des capacités, diminution des temps d'accès.
- évolution des logiciels avec l'apparition sur le marché de produits fiables et diversifiés.
- développement des systèmes en temps réel.
- modification dans l'approche des problèmes de gestion par les entreprises. (approche globale par des données organisées de façon rationnelle plutôt qu'approche par des étapes basées sur les fonctions à réaliser).

3) Rappel sur les systèmes de gestion de fichiers

Les données étant stockées sur des supports physiques (fichiers), les différents systèmes assurent des manipulations à 3 niveaux :

- support physique (exemple : une sauvegarde de fichiers constitue le passage des données d'un support à un autre)

- structures (opérations d'ouverture de fichier, fermeture, lecture, écriture)

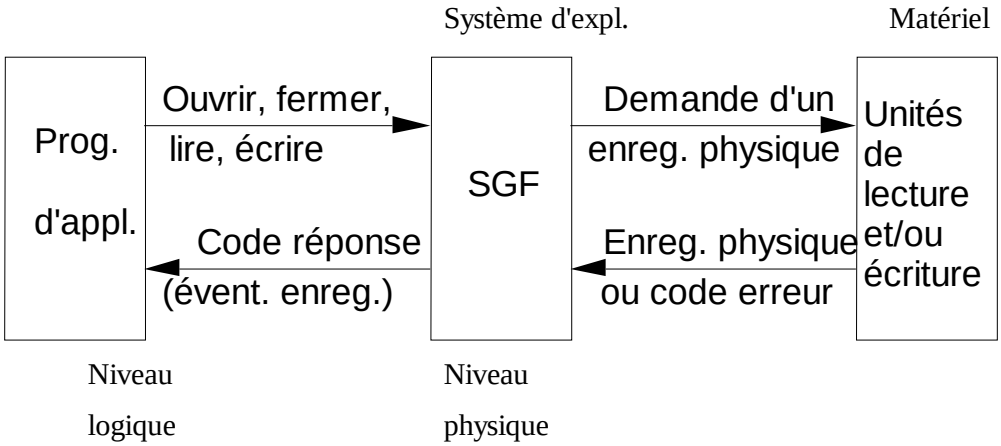
- contenu (calculs, tests...).

L'ensemble de ces fonctions est réalisé par:

* des programmes d'application,

* des unités de lecture et/ou d'écriture,

* des programmes de communication ou d'interface, l'ensemble de ces éléments constituant un S.G.F. (système de gestion de fichiers), que l'on peut représenter selon le schéma suivant



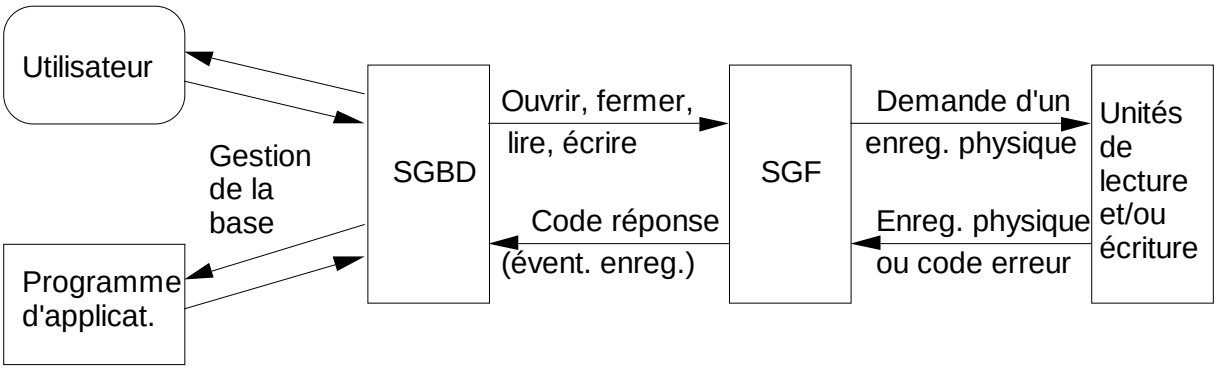
4) Système de gestion de base de données (SGBD)

C'est l'ensemble des programmes et des langages de commande (livrés par le constructeur ou la société qui commercialise le produit) qui permettent :

- la définition des différents "ensembles de données" de la base, et les relations existant entre eux,

- le traitement de ces données (interrogations, mises à jour, calculs, extractions...).

Le SGBD reçoit des commandes aussi bien des programmes d'application que des utilisateurs; il commande les manipulations de données, généralement par l'intermédiaire d'un SGF.



OBJECTIFS DE L'APPROCHE BDD

Pour pallier les inconvénients des méthodes classiques (fichiers), les bases de données tentent d'atteindre 4 objectifs principaux : intégration et corrélation, flexibilité (indépendance), disponibilité, sécurité.

La solution à ces problèmes passe par une distinction nette entre les données (avec la structure qui leur est associée), et les procédures de manipulation de ces données. Aux données, on associera une fonction d'administration des données, aux procédures de manipulation une fonction de programmation.

1) INTEGRATION ET CORRELATION

Dans les systèmes dits "classiques", chaque application gère son propre ensemble de données utiles, ses propres "fichiers": il existe donc un risque d'avoir une floraison de fichiers et de programmes disparates, redondants et non compatibles. Il en résulte plusieurs inconvénients :

- redondance d'où danger d'incohérence des données

la même donnée peut appartenir à plusieurs applications, induisant une déperdition de stockage. Toute modification de cette donnée est à enregistrer plusieurs fois : si cette mise à jour multiple n'est pas effectuée correctement, les données deviennent incohérentes. De plus, les mises à jour étant multiples, le coût de la mise à jour augmente du fait de la multiplication des entrées-sorties physiques.

- difficulté de mise en place de nouveaux traitements

mettre en place de nouvelles applications, non prévues a priori, entraîne dans un système classique, des duplications supplémentaires de données. De plus, pour les intégrer avec le reste des applicatifs en exploitation, des modifications, parfois importantes, sont nécessaires. Dans l'approche BDD, un "réservoir" commun (intégration) est constitué, représentant une modélisation (corrélation) aussi fidèle que possible de l'organisation réelle de l'entreprise. Toutes les applications puisent dans ce réservoir les données les concernant, évitant les duplications. A l'inverse, le partage des données communes par des utilisateurs ayant des besoins différents va poser de nouveaux problèmes de type contrôle de la concurrence d'accès.

2) FLEXIBILITE

La notion de flexibilité est essentielle dans la conception de tout système de taille importante. Dans le cas des BDD, cette notion porte généralement le nom d'indépendance.

Dans les systèmes classiques, tout changement intervenant dans le stockage des données (support, méthode d'accès physique...) entraîne des modifications lourdes (voire une reprogrammation complète !) des applications correspondantes. L'approche BDD poursuit trois objectifs, correspondant au concept d'indépendance des données par rapport aux traitements :

* indépendance physique: tout changement de support, de méthode d'accès reste transparent au niveau de l'utilisateur.

* indépendance logique : les programmes d'application sont rendus transparents à une modification dans l'organisation logique globale, par la définition de sous-schémas couvrant les besoins spécifiques en données.

* indépendance vis-à-vis des stratégies d'accès : de plus en plus, l'utilisateur n'a pas à prendre en charge l'écriture des procédures d'accès aux données. Il n'a donc pas à intégrer les modifications tendant à optimiser les chemins d'accès (ex: création d'index).

3) DISPONIBILITE

Le choix d'une approche BDD ne doit pas se traduire par des temps de traitement plus longs que ceux des systèmes antérieurs. En fait, tout utilisateur doit (ou devrait!) pouvoir ignorer l'existence d'utilisateurs concurrents.

L'aspect "performance" est donc crucial dans la mise en oeuvre d'une base de données. Un tel objectif ne peut être atteint que si la conception d'une base de données est menée de façon rigoureuse avec un découpage fonctionnel adéquat. Les règles et contraintes inhérentes sont évoquées lors de l'apprentissage d'une méthodologie d'analyse (exemple MERISE).

4) SECURITE

La sécurité des données (qui sera décrite en détail ultérieurement) est un terme générique qui recouvre plusieurs aspects:

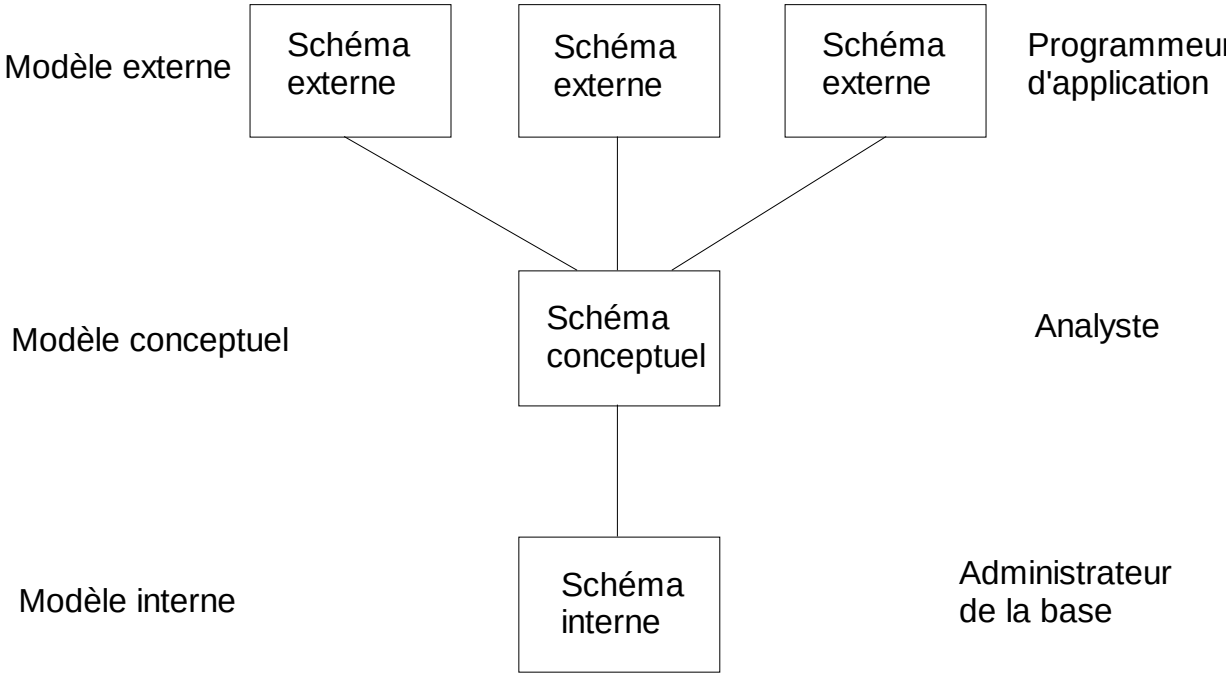
- l'intégrité, ou protection contre l'accès invalide (erreurs ou pannes), et contre l'incohérence des données vis-à-vis des contraintes de l'entreprise.

- la confidentialité, ou protection contre l'accès non autorisé ou la modification illégale des données.

Pour ne pas affecter les performances de façon sensible, la sécurité doit également être prise en compte dès la phase de conception.

ARCHITECTURE FONCTIONNELLE D'UN SGBD : ANSI-SPARC

Dans le cadre du groupe de normalisation nord américain (ANSI), un groupe d'études a été créé en 69, Standard Planning and Requirement Committee (SPARC), avec pour mission, une standardisation des SGBD. Les travaux ont abouti en 75 (ANSI75) par la proposition d'une architecture multi-niveaux : à chaque niveau fonctionnel, sont associés un modèle et un schéma de données, un langage de description de données (LDD) permettant de décrire les données du schéma, et un langage de manipulation de données (LMD) permettant de les utiliser (accès pour consultation, mise à jour...).



1) NIVEAU CONCEPTUEL

Il représente l'abstraction, aussi fidèle que possible, de l'univers de l'entreprise, après modélisation et indépendamment de toute référence à l'utilisation et à l'implantation en machine. Ainsi, le modèle conceptuel de données permet le passage d'un concret inaccessible (l'univers réel) à un abstrait manipulable : le schéma conceptuel. Celui-ci peut donc être considéré comme la description du contenu de la base; c'est le résultat d'un travail d'analyse et de conception d'un système d'information automatisé. La structure du système d'information est décrite de son point de vue général.

Un schéma conceptuel doit offrir les caractéristiques suivantes:

- puissance de représentation : aspects structurels, contraintes existant dans l'univers réel.

- stabilité et flexibilité : l'ajout d'une nouvelle donnée ou d'une nouvelle contrainte ne doit pas entraîner de changement important dans le schéma, ni d'"effet de bord".

- simplicité de compréhension : nombre d'éléments réduit, dissociation claire des différents concepts.

- simplicité d'utilisation : nombre restreint d'outils ou de primitives de manipulation.

- base formelle : la définition du schéma doit s'appuyer sur une méthode rigoureuse, mathématique, pour éviter toute ambiguïté d'interprétation et pour garantir la fiabilité des données.

Pour aboutir au schéma conceptuel, l'analyste doit repérer dans le réel, et recenser de manière exhaustive, toutes les entités et toutes les associations.

Une **entité** peut être définie comme une personne, un objet, un lieu, un statut, un évènement... qui ont une existence dans le monde réel. C'est un objet concret ou abstrait, possédant un certain nombre de caractéristiques spécifiques (exemple : le produit x coûte y francs). Généralement, les entités du monde réel se manifestent à travers des faits élémentaires.

Certains faits font intervenir plusieurs entités concernées, il apparaît la notion d'**association**. Une association (ou **lien**) est un ensemble de deux ou plusieurs entités, chacune d'elles jouant un rôle particulier.

Exemple : le fait que la voiture x appartienne à la personne y est une association entre les entités "voiture " et "personne y".

Selon la notation CODASYL, trois types de liens peuvent être envisagés :

- les liens fonctionnels notés N:1.

On a un lien N:1 de A vers B si toute occurrence de A détermine au plus une occurrence de B, et si, à toute occurrence de B, correspondent 0,1 ou plusieurs occurrences de A.

Exemple : dans une compagnie aérienne, connaissant le n° d'un vol, on en déduit d'une manière unique la destination, mais plusieurs vols peuvent avoir la même destination.

- les liens hiérarchiques notés 1:N.

On a un lien 1:N de A vers B si une occurrence de A permet de déterminer 0,1 ou plusieurs occurrences de B et si, à une occurrence de B, correspond au plus une occurrence de A.

Exemple : la polygamie est un lien 1:N de "homme" vers "femme".

- les liens maillés notés N:M.

Nous avons un lien maillé de A vers B s'il n'existe aucune restriction sur le nombre d'occurrences de A et B intervenant dans le lien.

Exemple : dans un lycée donné, un enseignant peut être amené à dispenser des cours dans plusieurs matières différentes; de la même façon, une matière donnée peut être dispensée par plusieurs enseignants.

Remarque : les cardinalités dans les MCD du formalisme MERISE ne correspondent pas forcément à cette représentation.

2) NIVEAU EXTERNE

Ce niveau logique comprend les "vues" spécifiques définies pour la manipulation des données ; il prend en compte les contraintes d'accès imposées par la nature des applications à considérer (indépendamment des caractéristiques techniques) ; il exprime les besoins en données des différents utilisateurs.

L'objectif à satisfaire à ce niveau est l'optimisation globale des accès par rapport aux applications ; d'autre part, la manipulation des données doit être très simple puisque l'utilisateur interagit au niveau externe.

Le modèle de données utilisé à ce niveau externe peut différer de celui utilisé au niveau conceptuel. Ainsi, certaines vues peuvent ne pas être construites dans la base, mais déduites par calcul à partir de certaines données du schéma conceptuel (exemple : ancienneté obtenue par différence entre année en cours et annnée d'embauche dans la société). Certaines associations entre données peuvent également être différentes, voire inversées.

3) NIVEAU INTERNE

Il correspond à la représentation en machine, aussi efficace que possible, du schéma conceptuel ; le schéma physique intègre les caractéristiques techniques (SGBD, matériel). L'efficacité doit tenir compte d'une part des contraintes d'implantation, d'autre part des critères d'utilisation.

FONCTIONNEMENT D'UN SGBD

La chronologie des opérations élémentaires effectuées lors d'une demande de donnée de la base par un programme est la suivante :

- un programme d'application A émet une demande de lecture d'une donnée (ou d'un groupe de données) de la base : la demande est transmise au SGBD.

- le SGBD traite la demande en consultant le sous-schéma (schéma externe) relatif au programme d'application A, obtenant ainsi la description des données.

- le système consulte le schéma logique global (schéma conceptuel) et détermine le type logique de données à extraire.

- le système examine la description physique de la base en rapport avec la requête logique et détermine le (ou les) enregistrement(s) physique(s) à lire.

- le système lance une commande au système d'exploitation pour rechercher physiquement l'enregistrement désiré.

- le système d'exploitation, par le biais de ses méthodes d'accès, accède à l'enregistrement physique.

- les données demandées sont transférées dans les buffers, ou mémoires tampons.

- le SGBD, à partir d'une comparaison entre schéma logique et sous-schéma A, extrait, des données stockées dans le buffer, l'enregistrement logique réclamé par le programme d'application. Il effectue également les transformations éventuelles de format.

- le SGBD transfère les données des buffers dans la zone de liaison du programme d'application A.

- le SGBD fournit également des informations "d'état" au programme d'application, lui signalant en particulier les erreurs éventuellement constatées au cours du processus d'extraction.

- le programme d'application, qui dispose des données et d'informations de "service" en assure la bonne exploitation!

Les ordres d'écriture dans la base physique sont traités par un processus similaire, toute modification ou adjonction étant en général précédée d'une opération de lecture.

A signaler que, dans la majorité des cas, le SGBD doit traiter simultanément plusieurs demandes de données en provenance de plusieurs programmes d'application, utilisant plusieurs schémas externes différents.

DIFFERENTS LANGAGES D'UN SGBD

1) Présentation

La présentation des SGBD réalisée dans les pages précédentes a fait apparaître la nécessité de bien différencier les deux étapes et objectifs différents que sont la définition des données a priori (rôle du DBA ou administrateur de la base) et leur utilisation a posteriori (par les utilisateurs ou les programmeurs d'application).

Le SGBD met donc à disposition deux types de langage :

Langage de Description de Données (LDD)

Il permet de décrire précisément la structure de la base et le mode de stockage des données. Tandis que l'utilisation de fichiers permet une description de données interne au programme, dans une approche Base de Données, on effectue la description de toutes les données une fois pour toutes : elle constitue l'ensemble des tables et dictionnaires de la base, son schéma (terminologie CODASYL).

En particulier, il précise la structure logique des données (nom, type, contraintes spécifiques...), la structure physique (mode d'implantation sur les supports, mode d'accès), la définition des sous-schémas ou "vues".

Langage de Manipulation de Données (LMD)

Il convient de rappeler que l'utilisation d'une BDD suppose un grand nombre d'utilisateurs (souvent non informaticiens) ayant tous des tâches et des besoins variés auxquels le LMD doit pouvoir répondre.

On peut répertorier :

- le langage d'interrogation ou langage de requête

  il évite le recours, en particulier pour des besoins ponctuels, à des langages généraux de programmation, qui impliquent des coûts prohibitifs et des délais souvent fort longs. Il doit être à syntaxe souple, accessible aux non-spécialistes et permettre la formulation de demandes utilisant des critères variés et combinés.

- le langage hôte

  pour les traitements réguliers ou mettant en oeuvre d'importants volumes d'informations, le SGBD doit pouvoir fournir un interface permettant l'utilisation de la base à l'aide des langages généraux (COBOL, PL/1, FORTRAN, ASSEMBLEUR...).

2) Classification des LMD

Les LMD se répartissent en 2 catégories principales :

- langages navigationnels (ex : SYMBAD)

on les rencontre avec les SGBD hiérarchiques ou réseaux. Les requêtes du langage (ou questions) décrivent les chemins d'accès aux différentes données, celles-ci étant généralement chaînées entre elles.

- langages algébriques (ex : SQL)

on les rencontre avec les SGBD relationnels. Ils utilisent, pour fournir des résultats aux requêtes, les opérateurs de l'algèbre relationnelle.

ROLE DE L'ADMINISTRATEUR DE LA BASE (DBA)

Rappelons succintement les différentes fonctions assumées par un SGBD:

description de la structure de la base (schéma)

organisation du stockage physique

manipulation des informations (sélection, extraction, mise à jour)

protection

pour personnaliser de façon fiable les accès à la base, il convient d'identifier l'utilisateur (code et mot de passe) et de vérifier qu'il est autorisé à effectuer sur les données les traitements qu'il demande (contrôle des droits d'accès).

sécurité, restauration (possibilité de reconstituer la base dans un état satisfaisant après tout incident)

optimisation des ressources

le logiciel doit fournir des statistiques précises sur l'état de la base et permettre des réorganisations physiques périodiques qui éviteront la dégradation des performances globales du système

intégrité des données (cohérence des informations les unes par rapport aux autres).

L'essentiel de la mise en oeuvre de ces fonctions revient à une personne (ou une équipe) appelée **administrateur** de la BDD. Elle dispose d'un ensemble d'outils logiciels pour l'assister dans sa tâche. Son rôle consiste à :

intervenir en tant que conseil lors de l'étape conceptuelle de l'analyse (responsabilité de gestion des données)

décider des techniques d'accès et de l'implantation physique

gérer les diverses autorisations d'accès

définir les stratégies de reprise en cas d'incident

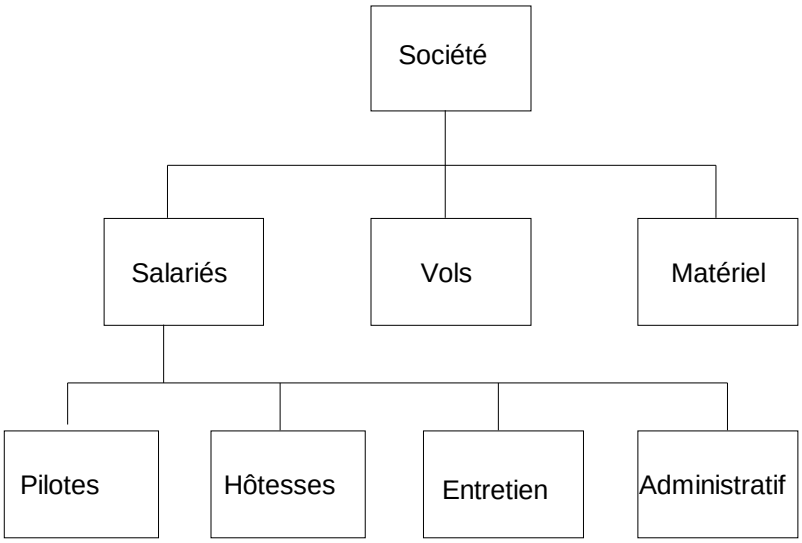
suivre régulièrement les performances du système et réaliser en conséquence les modifications ou évolutions qui s'imposent.

PRESENTATION DES PRINCIPAUX MODELES LOGIQUES

Un très grand nombre de modèles existent sur le marché, réalisant plus ou moins bien les diverses fonctions évoquées précédemment. Les 3 principaux, par le nombre d'installations, sont, dans l'ordre chronologique de leur arrivée sur le marché, le modèle hiérarchique, le modèle réseau (ou navigationnel), le modèle relationnel.

1) LE MODELE HIERARCHIQUE

Exemple : représentation (partielle!) du SI d'une compagnie aérienne



L'ancêtre, et le plus répandu, en est le SGBD IMS (Information Management System), développé et commercialisé par IBM.dans les années 70; on peut également citer le système S2000 (1978).

Sa caractéristique principale est la très forte dépendance entre la description de la structure des données et la manière dont celle-ci se trouvent enregistrées sur le support physique. Les éléments de base du modèle sont des enregistrements logiques reliés entre eux pour constituer un arbre ordonné ou arborescence valuée.

Les différentes entités (ou segments) constituent les noeuds, celui de plus haut niveau portant le nom de racine; les branches (pointeurs logiques entre entités) constituent les liens. Chaque segment est une collection d'objets appelés champs (ou fields).

Chaque noeud, sauf la racine, a un seul arc incident du type 1:N et un (ou plusieurs) arc(s) émergeant(s). Plus simplement, chaque segment a obligatoirement un père, mais un seul, et peut avoir plusieurs fils.

De façon visuelle, cette représentation schématique fait apparaître:

- les avantages du modèle

rigueur des structures et des chemins d'accès, simplicité relative de l'implémentation, adéquation parfaite du modèle à une entreprise à structure arborescente.

- les contraintes un peu lourdes qui en sont la contrepartie

* les accès se font normalement depuis la racine et uniquement depuis la racine

* la structure élude la possibilité de liens N:M, ne permettant que le lien 1:N. La représentation d'autres relations impose de ce fait une redondance de l'information.

Exemple: comment représenter un parc de véhicules et un ensemble de chauffeurs, chaque chauffeur pouvant conduire plusieurs véhicules, et un véhicule pouvant être conduit par plusieurs chauffeurs ?

* les "anomalies" que l'on est amené à constater lors des opérations de mise à jour (on parle de stockage : insertion, destruction, modification). Ainsi, l'élimination d'un noeud entraîne l'élimination de tous les segments de niveau inférieur qui lui sont rattachés (risque de perdre des données uniques)

* l'indépendance logique très réduite : la structure du schéma doit refléter les besoins des applications.

* de plus, il n'existe pas d'interface utilisateur simple.

2) LE MODELE EN RESEAU

Il peut être considéré comme une évolution du modèle hiérarchique, en lui intégrant les résultats du travail du groupe CODASYL (comité de langage de programmation), lequel avait démarré l'étude d'une extension de COBOL pour manipuler les bases de données. En 1969, il donne ses premières recommandations concernant syntaxe et sémantique du LDD et du LMD.

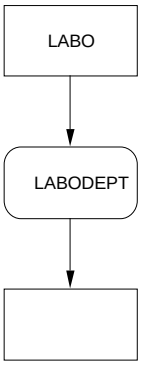
Même si cette vue est un peu simplificatrice, une base en réseau peut être décrite comme un certain nombre de fichiers comportant des références des uns vers les autres. Les entités sont connectées entre elles à l'aide de pointeurs logiques. Ce concept permet d'associer, à un enregistrement d'un ensemble de données A, une série d'enregistrements (ou records) d'un autre ensemble de données B. On constitue ainsi des SET, ou COSET, structure fondamentale du modèle en réseau :

- le lien entre les enregistrements de A et ceux de B est 1:N

- le COSET comporte un type d'enregistrement "propriétaire" (l'enregistrement de A est dit OWNER) et un type d'enregistrement "membre" (les enregistrements de B sont MEMBER).

Supposons l'exemple suivant : un institut de recherche regroupe plusieurs laboratoires, divisés en départements, chacun de ceux-ci employant plusieurs chercheurs.

Représentation : diagramme de Bachman.

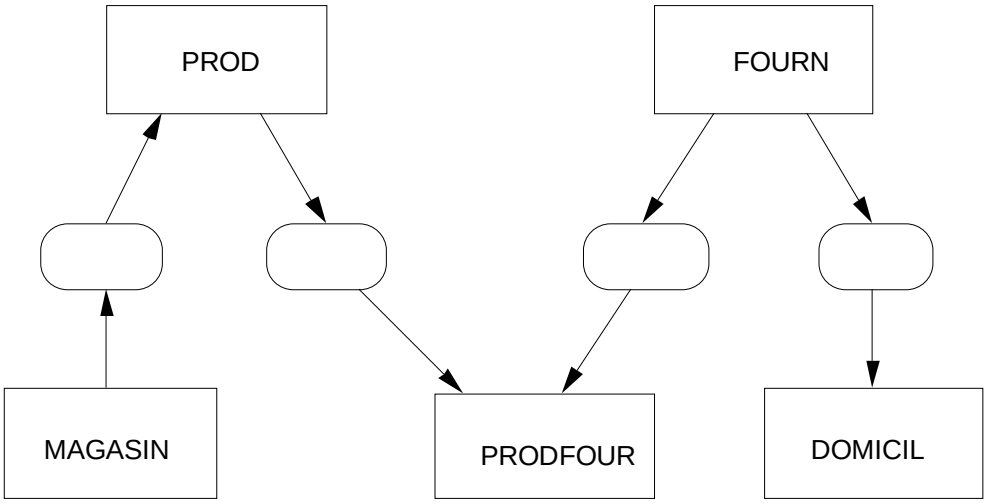


On ne rencontre aucune restriction à la conception du modèle : un type de "record" peut à la fois être propriétaire et membre vis-à-vis de plusieurs sets ; de même, il peut être propriétaire (ou membre) de plusieurs sets.

Exemple :

schéma représentant le sous-système d'information

produits / magasins de stockage / fournisseurs / domiciliations bancaires



Avantages et inconvénients du modèle

* représentation naturelle des liens maillés N:M

* absence d'anomalies pour les opérations de stockage

* commercialisation importante des systèmes correspondants (DMS, IDMS, TOTAL, IDS II, SOCRATE...), mais

- pas d'indépendance par rapport aux stratégies d'accès

- procéduralité importante des langages de manipulation ; l'utilisateur doit "naviguer" dans le réseau logique constitué par les enregistrements et les chaînes de pointeurs.

3) LE MODELE RELATIONNEL

C'est un article publié en 1969 par un mathématicien du centre de recherche IBM, Codd, qui définit les bases de ce modèle relationnel.

Codd s'est intéressé au concept d'information et a cherché à le définir sans se préoccuper de la technique informatique, de ses exigences et de ses contraintes. Il a étudié un modèle de représentation des données qui repose sur la notion mathématique de "relation". Dans la pratique, une relation sera représentée par une table de valeurs.

Exemple: représentation d'une table du personnel

Théorie et définitions

- Une relation est un ensemble de tuples ou uplets (lignes ou articles), dont l'ordre est sans importance; chaque tuple est unique. Les colonnes de la table sont appelées attributs ou champs; leur ordre est défini lors de la création de la table.
- Une clé est un ensemble ordonné d'attributs qui caractérise un tuple. Une clé primaire le caractérise de manière unique, à l'inverse d'une clé secondaire.
- On dit qu'un attribut A est un déterminant si sa connaissance détermine celle de l'attribut B.

Objectifs du modèle

- proposer des schémas de données faciles à utiliser
- améliorer l'indépendance logique et physique
- mettre à disposition des utilisateurs des langages de haut niveau pouvant éventuellement être utilisés par des non-informaticiens.
- optimiser les accès à la base de données
- améliorer l'intégrité et la confidentialité
- prendre en compte une variété d'applications
- fournir une approche méthodologique dans la construction des schémas.

Caractéristiques du modèle

(DB2, INGRES, ORACLE, FOCUS, NOMAD, DATACOM)

* les structures de données sont simples : ce sont des tables à deux dimensions où les éléments sont des données élémentaires.

* un ensemble d'opérateurs (union, intersection, différence, produit cartésien, projection, sélection, jointure, division) , l'algèbre relationnelle, appliqués aux relations, permet la définition, la recherche et la mise à jour des données. Cette algèbre est la base des langages de manipulation non procéduraux, type SQL.

* un ensemble de contraintes d'intégrité (unicité de clé, contrainte référentielle) définit l'état cohérent de la base.

* toute information de la base de données est représentée par des valeurs dans des tables.

* il n'y a pas de pointeurs visibles par l'utilisateur entre tables.

Avantages et inconvénients

- indépendance de l'utilisateur vis-à-vis de la structure logique, de la structure physique et des stratégies d'accès, d'où simplicité et clarté dans l'utilisation. puissance et uniformité de la représentation.

très grande souplesse d'expression dans la sécurité des données.

existence d'interfaces non procédurales.

indépendance des programmes vis-à-vis de la base : une modification du schéma ne nécessite pas de modification des applicatifs existants.

- problèmes de performance pour certains modèles

difficulté de conception d'un schéma conceptuel propre (mise en application des règles de normalisation), étape indispensable à un fonctionnement satisfaisant du modèle.

LE MODELE RELATIONNEL

1. INTRODUCTION : LES OBJECTIFS DU MODELE

Le modèle relationnel a été introduit par E. F. Codd [Codd70] qui travaillait au fameux Centre de Recherche d'IBM San-José. La première volonté du modèle relationnel fut d'être un modèle ensembliste simple, supportant des ensembles d'enregistrements aussi bien au niveau de la description que de la manipulation. Les premières idées d'un modèle ensembliste avaient été proposées un peu avant, notamment dans [Childs68]. Le modèle relationnel est aujourd'hui la base de nombreux systèmes, et les architectures permettant d'accéder depuis une station de travail à des serveurs de données s'appuient en général sur le modèle relationnel. Le relationnel a donc atteint ses objectifs au-delà de toute espérance.

Les premiers objectifs du modèle ont été précisés par E. F. Codd [Codd7O] comme suit

1). Permettre un haut degré d'indépendance des programmes d'applications et des activités interactives à la représentation interne des données, en particulier aux choix des ordres d'implantation des données dans les fichiers, des index et plus généralement des chemins d'accès.

2) Fournir une base solide pour traiter les problèmes de cohérence et redondance des données.

Ces deux objectifs qui n'étaient pas atteints par les modèles réseau et hiérarchique, ont été pleinement satisfaits par le modèle relationnel, d'une part grâce à la simplicité des vues relationnelles qui permettent de percevoir les données sous forme de tables à deux dimensions et d'autre part grâce aux règles d'intégrité supportées par le modèle et ses fondements logiques.

En addition aux objectifs fixés par E. F. Codd, le modèle relationnel a atteint un troisième objectif:

3) Permettre le développement de langages de manipulation de données non procéduraux basés sur des théories solides.

Ce troisième objectif est atteint d'une part à l'aide de l'algèbre relationnelle qui permet de manipuler des données de manière très simple et formelle comme les opérateurs arithmétiques permettent de manipuler des entiers et d'autre part à l'aide des langages assertionnels basés sur la logique qui permettent de spécifier les données que l'on souhaite obtenir sans dire comment les obtenir

Finalement, le modèle relationnel a atteint deux autres objectifs non prévu initialement:

4). Etre un modèle extensible permettant de modéliser et de manipuler simplement des données tabulaires, mais pouvant être étendu pour modéliser et manipuler des données complexes.

5) Devenir un standard pour la description et la manipulation des bases de données.

L'objectif 4 est très important, car il a permis d'intégrer de nouveaux concepts au modèle relationnel, notamment la plupart des concepts de l'orienté objets. Les premiers travaux de recherche dans ce sens ont été effectués par Codd lui-même [Codd79]. puis poursuivis à Bell Laboratories [Zaniolo83], à Berkeley [Stonebraker8l] et, en France, à l'INRIA [Gardarin89].

L'objectif 5 a été réalisé grâce en particulier à IBM : le modèle relationnel et son langage SQL ont été normalisé au niveau international en 1986 [1S089].

Dans ce chapitre, nous allons tout d'abord présenter les concepts structuraux de base du modèle relationnel qui permettent de modéliser les données sous forme de tables à deux dimensions. Nous soulignerons ensuite les règles de cohérence des relations qui sont considérés comme partie intégrante du modèle. Puis nous introduirons l'algèbre relationnelle, qui est l'outil formel indispensable afin de manipuler des relations. Les nombreuses extensions de cette algèbre seront également présentées. En conclusion, nous démontrons les vastes possibilités d'enrichissement offertes par le modèle relationnel, possibilités qui seront étudiées plus en détail au niveau des modèles objets et logiques, sujets des chapitres VII et VIII de ce livre.

2. LES STRUCTURES DE DONNEES DE BASE

2.1. DOMAINE, ATTRIBUT ET RELATION

Le modèle relationnel est basé sur la théorie mathématique bien connue des relations. Cette théorie se construit à partir de la théorie des ensembles. Trois notions de base sont importantes pour introduire les bases de données relationnelles. La première notion permet de définir les ensembles de départ. Ces ensembles sont les domaines de valeurs.

> Notion V.1 : Domaine (Domain)
> Ensemble de valeurs caractérisé par un nom.

Les domaines sont donc les ensembles dans lesquels les données prennent valeur. Comme un ensemble, un domaine peut être défini en extension, en donnant la liste des valeurs composantes, ou en intention, en définissant une propriété caractéristique des valeurs du domaine. Au départ, les domaines ENTIER, REEL, BOOLEEN, CARACTERES (une chaîne de caractères de longueur fixe ou variable) sont définis en intention. A partir de ces domaines, il est possible de définir en intention des domaines plus spécifiques tels que MONNAIE (réel avec 2 chiffres derrière la virgule), DATE (entier de 6 chiffres jour, mois et an), TEMPS (heure en seconde), etc. Un domaine peut toujours être défini en extension, par exemple le domaine des couleurs de vins COULEUR-VINS **=** {Rosé, Blanc, Rouge}.

Rappelons que le produit cartésien d'un ensemble de domaines Dl, D2,.., Dn est l'ensemble des vecteurs < vi, v2, ..., vn > où, pour i variant de 1 à n. vi est une valeur de Di. Par exemple, le produit cartésien des domaines COULEUR-VINS **=** {ROSE, BLANC, ROUGE} et CRUS **=** {VOLNAY, SANCERRE, CHABLES} est composé des neuf vecteurs représentés figure V.1.

COULEUR-VINS = {ROSE, BLANC, ROUGE}
CRUS = {VOLNAY, SANCERRE, CHABLIS}

ROSE	VOLNAY
ROSE	SANCERRE
ROSE	CHABLIS
BLANC	VOLNAY
BLANC	SANCERRE
BLANC	CHABLIS
ROUGE	VOLNAY
ROUGE	SANCERRE
ROUGE	CHABLIS

Figure V.1 : Exemple de produit cartésien

Nous pouvons maintenant introduire la notion de relation, bien sûr à la base du modèle.

> Nation V.2 : Relation (Relation)
> Sous-ensemble du produit cartésien d'une liste de domaines caractérisé par un nom.

Etant un sous-ensemble d'un produit cartésien, une relation est composée de vecteurs. Une représentation commode d'une relation sous forme de table à deux dimensions a été retenue par Codd et est généralement utilisée. Chaque ligne correspond à un vecteur alors que chaque colonne correspond à un domaine du produit cartésien considéré un même domaine peut bien sûr apparaître plusieurs fois. Par exemple, à partir des domaines COULEURS_VINS et CRUS, il est possible de composer la relation COULEURS_CRUS représentée sous forme tabulaire figure V.2.

COULEURS_CRUS

Couleur	Cru
ROSE	SANCERRE
ROSE	CHABLIS
BLANC	SANCERRE
ROUGE	VOLNAY
ROUGE	SANCERRE
ROUGE	CHABLIS

Afin de pouvoir distinguer les colonnes d'une relation sans utiliser un index, et ainsi de rendre leur ordre sans importance tout en permettant plusieurs colonnes de même domaine, il est nécessaire d'associer un nom à chaque colonne. Une colonne se distingue d'un domaine en ce sens qu'elle prend valeur dans un domaine et peut à un instant donné comporter seulement certaines valeurs du domaine. Par exemple, si le domaine est l'ensemble des entiers, seulement quelques valeurs seront prises à un instant donné, par exemple {10, 20, 30}. L'ensemble des valeurs d'une colonne de relation est en général fini. Afin de bien distinguer domaine et colonne, on introduit la notion *d'attribut* comme suit.

> Notion V.3 : Attribut (attribute)
> Colonne d'une relation caractérisée par un nom.

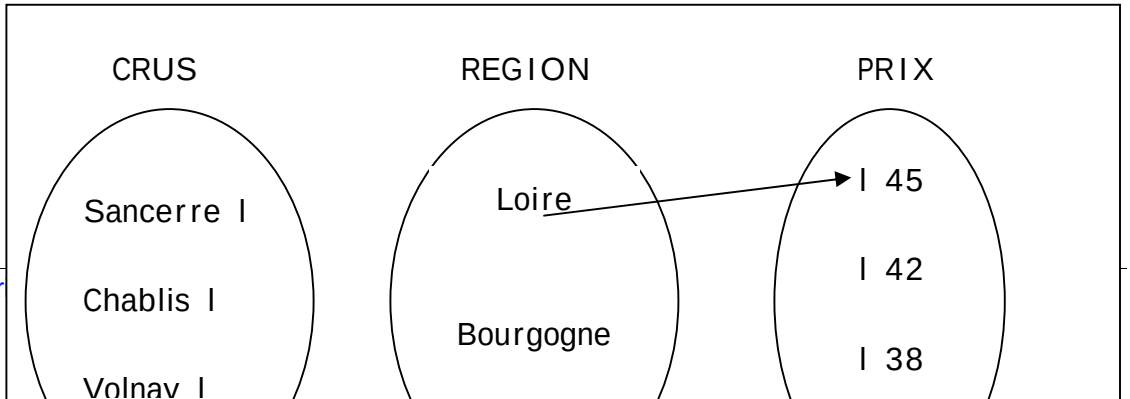
Le nom associé à un attribut est souvent porteur de sens. Il est donc en général différent de celui du domaine qui supporte l'attribut. Ainsi, par exemple, la première colonne de la relation COULEUR-CRUS pourra être appelée simplement COULEUR et la deuxième CRU. COULEUR et CRU seront donc deux attributs de la relation COULEUR-CRUS.

Les lignes d'une relation correspondent à des n-uplets de valeurs. Une ligne est aussi appelée t*uple* (du mot anglais tuple) : un tuple correspond en fait à un enregistrement dans une relation (encore appelée table).

> Notion V.4 : Tuple (tuple)
> Ligne d'une relation correspondant à un enregistrement.

La notion de relation est bien connue en mathématiques : une relation n-aire est un ensemble de n-uplets. Un cas particulier souvent employé est la relation binaire qui est un ensemble de paires ordonnées. Une relation n-aire est souvent représentée par un graphe entre les ensembles composants. Ainsi, la relation LOCALISATION, donnant la région de chaque cru et certains prix moyens des vins associés est représentée par un graphe figure V.3.

Une relation peut aussi être représentée sur un diagramme à n dimensions dont les coordonnées correspondent aux domaines participants. Ainsi, la relation STATISTIQUE, d'attributs AGE et SALAIRE, donnant la moyenne des salaires par âge est représentée figure V.4.



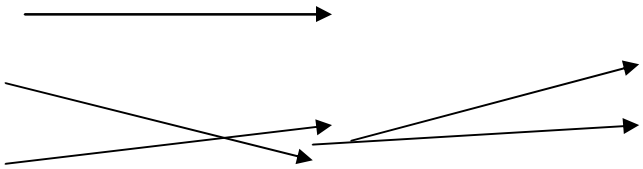
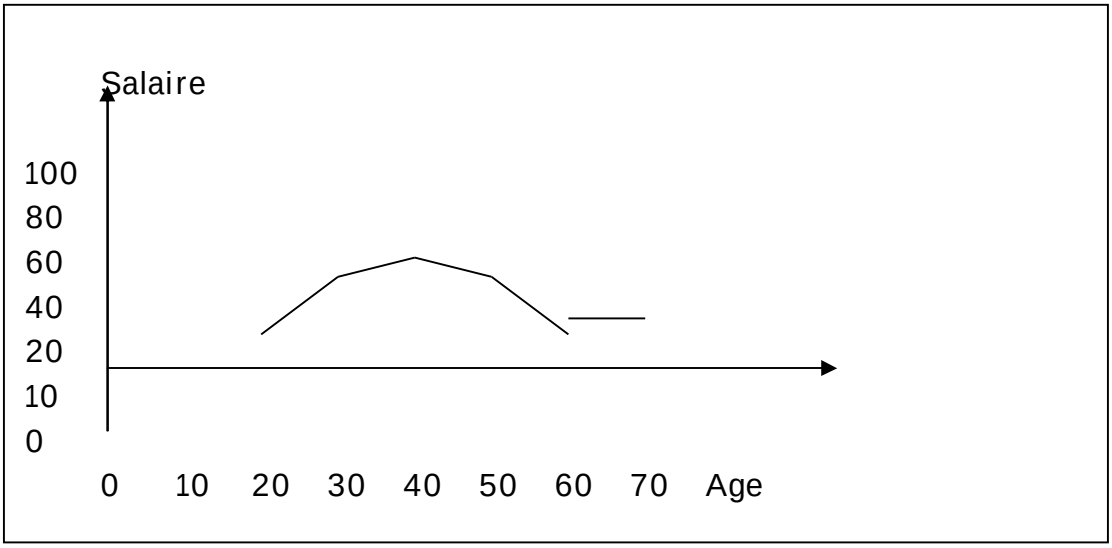


Figure V.3 : Graphe de la relation LOCALISATION



Une autre perception mathématique d'une relation consiste a voir chaque tuple t comme une fonction : t: Ai à Di pour i = 1, 2 ... n qui applique donc les attributs sur les domaines. Ainsi, une relation peut être vue comme un ensemble fini de fonctions. Bien sûr, cet ensemble varie en fonction du temps. Tout ceci montre la diversité des outils mathématiques qui peuvent être appliqués aux relations. En conséquence, le modèle relationnel peut apparaître comme un modèle mathématique simple des données. De plus, grâce à la représentation tabulaire il est simple à appréhender pour les non mathématiciens.

2.2. EXTENSIONS ET INTENTIONS

Comme tous les modèles de données, le modèle relationnel permet de décrire des données dont les valeurs varient en fonction du temps. En particulier, les relations varient en fonction du temps, en ce sens que des tuples sont ajoutés, supprimés et modifiés dans une relation au cours de sa vie. Cependant, la structure d'une relation caractérisée par les trois concepts de domaine, relation et attribut est un invariant pour la relation qui ne change pas en fonction du temps. Cette structure est capturée dans le *schéma de la relation.*

> Notion 5 : Schéma de Relation (Relation Schema)
> Nom de la relation suivi de la liste dos attributs et de la définition de leurs domaines.

Un schéma de relation est noté sous la forme :

R (A1: D1, A2: D2, ..., Ai: Di, ...,An: Dn)

où R est le nom de la relation, Ai les attributs et Di les domaines associés. A titre d'exemple, nous donnons figure V.5 deux schémas de relations celui de la relation LOCALISATION introduite ci-dessus et celui de la relation VINS qui représente des vins caractérisés par un numéro, un cru, un millésime et un degré. Les domaines utilisés sont ceux des entiers, des réels et des chaînes de caractères de longueur variable (CHARVAR). La plupart du temps, lorsqu'on définit le schéma d'une relation, les domaines sont implicites et découlent du nom de l'attribut aussi, on omet de les préciser.

> LOCALISATION (CRU: CHARVAR, REGION: CHARVAR, PRIX: REEL)
> VINS (NV:ENTIER, CRU: CHARVAR, MILL:ENTIER, DEGRE: REEL)

Figure V.5 : Schémas des relations LOCALISATION et VINS

Il est à souligner que le schéma d'une relation représente son *intention,* c'est-à-dire les propriétés (ou au moins certaines) communes et invariantes des tuples qu'elle va contenir au cours du temps. Au contraire, une table représente une *extension* d'une relation (voir figure V.2 par exemple), c'est-à-dire une vue des tuples qu'elle contient à un instant donné. Une extension d'une relation R est aussi appelée *instance* de R. L'intention est le résultat de la description des données, alors qu'une extension (ou instance) fait suite à des manipulations et représente un état de la base.

3. LES REGLES D'INTEGRITE STRUCTURELLE

Les règles d'intégrité sont les assertions qui doivent être vérifiées par les données contenues dans une base. Il est possible de distinguer les règles structurelles qui sont inhérentes au modèle de données, c'est-à-dire nécessaires à sa mise en œuvre, et les règles de comportement propres au schéma particulier d'une application. Le modèle relationnel impose a priori une règle minimale qui est l'unicité des clés comme nous allons le voir ci-dessous. Il est commode et courant [Date81] d'ajouter trois types de règles d'intégrité supplémentaires — les contraintes de références, les contraintes d'entité et les contraintes de domaine — cela afin d'obtenir les règles d'intégrité structurelles supportées par le modèle relationnel.

3.1 UNICITE DE CLE

Par définition, une relation est un ensemble de tuples. Un ensemble n'ayant pas d'élément en double, il ne peut exister deux fois le même tuple dans une relation. Afin d'identifier les tuples d'une relation sans donner toutes les valeurs et d'assurer simplement l'unicité des tuples, la notion de *clé* est utilisée.

> Notion V.6 : Clé (Key)
> Ensemble d'attributs minimal dont la connaissance des voleurs permet d'identifier un Iup!e unique de la relation considérée.

De manière plus tonnelle, une clé d'une relation R est un ensemble d'attributs K tel que, quels que soient les tuples t1 et t2 d'une instance de R, t1(K) ≠ t2(K), c'est-à-dire que t1 et t2 ont des valeurs de K différentes. Un ensemble d'attributs contenant une clé est appelée *super-clé* [Ullman88].

Toute relation possède au moins une clé car la connaissance de tous les attributs permet d'identifier un tuple unique. Dans le cas où il existe plusieurs clés, on en choisit en général une de manière arbitraire qui est appelée *clé primaire.* Par exemple, NV peut constituer une clé primaire pour la relation VINS. Le couple <CRU, MIILLESIME> est une clé alternative.

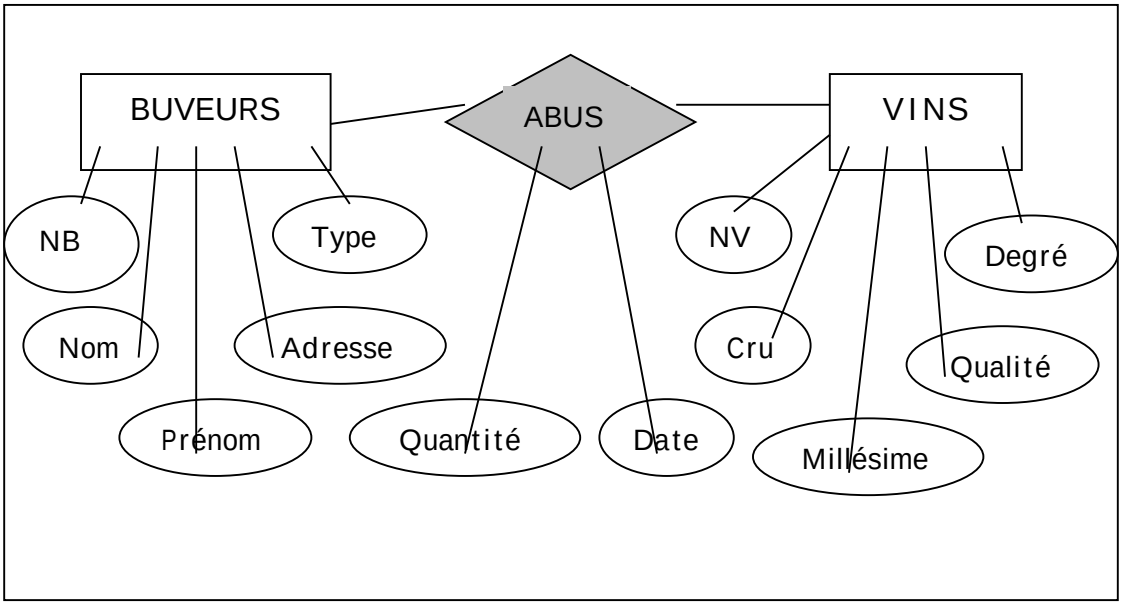
Il est à souligner que la notion de clé caractérise l'intention d'une relation dans toute extension possible d'une relation, il ne peut exister deux tuples ayant même valeur pour les attributs clés. La détermination d'une clé pour une relation nécessite donc une réflexion sur la sémantique de la relation, c'est-à-dire sur toutes les extensions possibles et non pas sur une extension particulière.

3.2. CONTRAINTES DE REFERENCES

Le modèle relationnel est souvent utilisé afin de représenter des entités du monde réel qui sont les objets ayant une existence propre, et des associations

entre ces objets [Chen76]. Une entité correspond alors à un tuple dans une relation qui comporte à la fois la clé de l'entité et ses caractéristiques sous forme d'attributs. Une entité est identifiée par la valeur de sa clé. Un type d'association est modélisé par une relation comportant les clés des entités participantes ainsi que les caractéristiques propres à l'association.

A titre d'exemple, nous considérons les entités BUVEURS et VINS du monde réel : la clé de l'entité BUVEURS est le numéro de buveur NB et celles de l'entité VINS le numéro de vin NV. Ces entités peuvent être associées par une consommation d'un vin par un buveur : une consommation sera par exemple modélisée par une association ABUS entre le buveur et le vin. Cette base typique et simple, qui servira souvent d'exemple, est appelée DEGUSTATION. Le diagramme entité association de cette base est représenté figure V.6. En relationnel, chaque entité est représentée par une table. Une association est aussi représentée par une table, dont les attributs seront les clés des entités participantes, c'est-à-dire NB et NV, ainsi que les attributs caractérisant l'association, par exemple la date et la quantité bue. On aboutit ainsi au schéma relationnel représenté figure V.7.



BUVEURS (NB, NOM, PRENOM, ADRESSE, TYPE)
VIN (NV, CRU, MILL QUALITE, DEGRE)
ABUS (NB, NV, DATE, QUANTITE)

L'utilisation du modèle relationnel pour représenter des entités et des associations ne permet pas jusque là de représenter le fait que l'association entre une consommation et un vin est obligatoire, c'est-à-dire que tout abus doit correspondre à un vin existant dans la base. De même, le fait que le lien entre

une consommation et un buveur soit obligatoire est perdu. Le maintient de liens obligatoires a justifié l'introduction de la notion de contrainte d'intégrité référentielle [Date8l].

> Notion V.7 : Contrainte référentielle (Referential constraint)
> Contrainte d'intégrité portant sur une relation R1, consistant à imposer que la valeur d'un groupe d'attributs apparaisse comme valeur de clé dans une autre relation R2.

Une telle contrainte d'intégrité s'applique en général aux associations obligatoires : une telle association ne peut exister que si les entités participant à l'association existent déjà. A noter que dans la définition, R1 et R2 ne sont pas forcément distinctes : l'association peut en effet porter sur deux entités de même type, par exemple entre une personne et ses parents.

La représentation de contraintes de référence peut s'effectuer par la définition de *clés étrangères* dans une relation : une clé étrangère est un groupe d'attributs qui doit apparaître comme clé dans une (autre) relation. Par exemple, la relation ABUS (NB, NV, DATE, QUANTITE)
a pour clé <NB, NV, DATE> et a deux clés étrangères NB dans BUVEURS et NV dans VINS.

Les contraintes référentielles définissent des liens obligatoires entre relations. Ce sont des contraintes très fortes qui impactent les opérations de mises à jour. Lors de l'insertion d'un tuple dans une relation référençant, il faut vérifier que les valeurs de clés étrangères existent dans les relations référencées. Par exemple, lors de l'insertion d'un abus, il faut que le vin et le buveur associés existent, sinon l'insertion est refusée pour cause de violation d'intégrité. Lors de la suppression d'un tuple dans une relation référencée, il faut vérifier qu'aucun tuple n'existe dans chaque relation référençante s'il en existe, le système peut soit refuser la suppression, soit cascader La suppression (c'est-à-dire, supprimer les tuples référençants). Par exemple, la suppression d'un vin entraînera la vérification qu'aucun abus ne référence ce vin. Les contraintes d'intégrité référentielles sont donc des liens forts qui rendent les relations dépendantes ; en quelque sorte, elles introduisent des liens hiérarchiques depuis les relations référencées vers les relations référençantes.

3.3. VALEURS NULLES ET CLES

Lors de l'insertion de tuples dans une relation, il arrive fréquemment qu'un attribut soit inconnu ou non applicable par exemple, la quantité de vin bu par un buveur à une certaine date peut être inconnue ; si un vin ne possède pas de millésime, l' attribut millésime est inapplicable à ce vin. On est alors amené à introduire dans la relation une valeur conventionnelle, appelée *valeur nulle.*

> Notion V.8 : Valeur nulle (Null value)
> Valeur conventionnelle introduite dans une relation pour représenter une information inconnue ou inapplicable.

La signification précise d'une valeur nulle est souvent ambiguë ; le groupe ANSI/X3/SPARC a recensé quatorze significations possibles, parmi lesquelles valeur inconnue et valeur inapplicable sont les plus caractéristiques.

Tout attribut dans une relation ne peut prendre une valeur nulle ; en effet, l'existence d'une clé unique impose la connaissance de la clé afin de pouvoir vérifier que cette valeur de clé n'existe pas déjà. Ainsi, une contrainte structurelle du modèle relationnel est la contrainte d'entité définie ci-dessous [Date81].

> Notion V.9 : Contrainte d'entité (Entity constraint)
> Contrainte d'intégrité imposant que toute relation possède une clé primaire et que tout attribut participant à cette clé primaire soit non nul.

A moins qu'il n'en soit spécifié autrement par une contrainte sémantique, le modèle relationnel n'impose pas que les clés étrangères qui n'appartiennent pas à une clé primaire soient non nulles. Cela peut permettre une certaine souplesse, par exemple d'enregistrer des employés qui ne sont attachés à aucun service.

3.4. CONTRAINTES DE DOMAINES

En théorie, une relation est construite à partir d'un ensemble de domaines. En pratique, les domaines gérés par les systèmes sont souvent limités aux types de base entier, réel, chaîne de caractères, parfois monnaie et date. Afin de spécialiser un type de données pour composer un domaine plus fin (par exemple, le domaine des salaires mensuels qui sont des réels compris entre 5 000 et 1 000 000 de francs), la notion de *contrainte d'intégrité de domaine* est souvent ajoutée aux règles d'intégrité structurelle du relationnelle. Cette notion peut être introduite comme suit.

> Notion V.1O : Contrainte de domaine (Domain constraint)
> Contrainte d'intégrité imposant qu'une colonne d'une relation doit comporter des voleurs vérifiant une assertion logique.

L'assertion logique est soit l'appartenance à une plage de valeurs ou à une liste de valeurs. Par exemple, SALAIRE ≥ 5 000 et ≤ 1 000 000,
COULEUR ε {BLEU, BLANC, ROUGE}, etc. Les contraintes permettent de contrôler la validité des valeurs introduites lors des insertions ou mises à jour. La non-nullité d'une colonne peut aussi être considérée comme une contrainte

de domaine, par exemple DEGRE ≠ NULL.

PROPOSITION DE SOLUTION

DU CAS

JEUX OLYMPIQUES D'HIVERS

CAHIER DES CHARGES

Les Organisateurs des jeux Olympique d'hiver souhaitent s'équiper d'un système informatique capable de gérer les épreuves , les concurrents et les Responsables.

Les personnes responsables des jeux se sont organisées de façon hiérarchique. Au sommet de la pyramide se trouve un "Responsable générale" dont dépendent des responsables de discipline (Ski alpin , patinage , Bobsleigh ..). De ces derniers dépendent à leur tour des responsables moins importants ,et ainsi de suite jusqu'à la personne de base.

Tout le monde a un rôle dans l'organisation des jeux : "juge a l'arrivée , chronométreur , juge de parcours…. " et pour des raisons pratiques sera identifié par un N°Matricule. On gèrera aussi leur nom et leur coordonnées.
Chaque épreuve a lieu dans une des stations de la région. Une même station dont on connaît le nom et l'altitude peut accueillir plusieurs épreuves mais à des jours différents et du même type. A l'organisation d'une épreuve sont affectés plusieurs responsables .

Les responsables peuvent s'occuper de plusieurs épreuves. Ces dernières ont un Code alphabétique . Chaque épreuve fait partie d'une discipline et d'une seule. Elles se déroulent quelquefois en plusieurs manches.

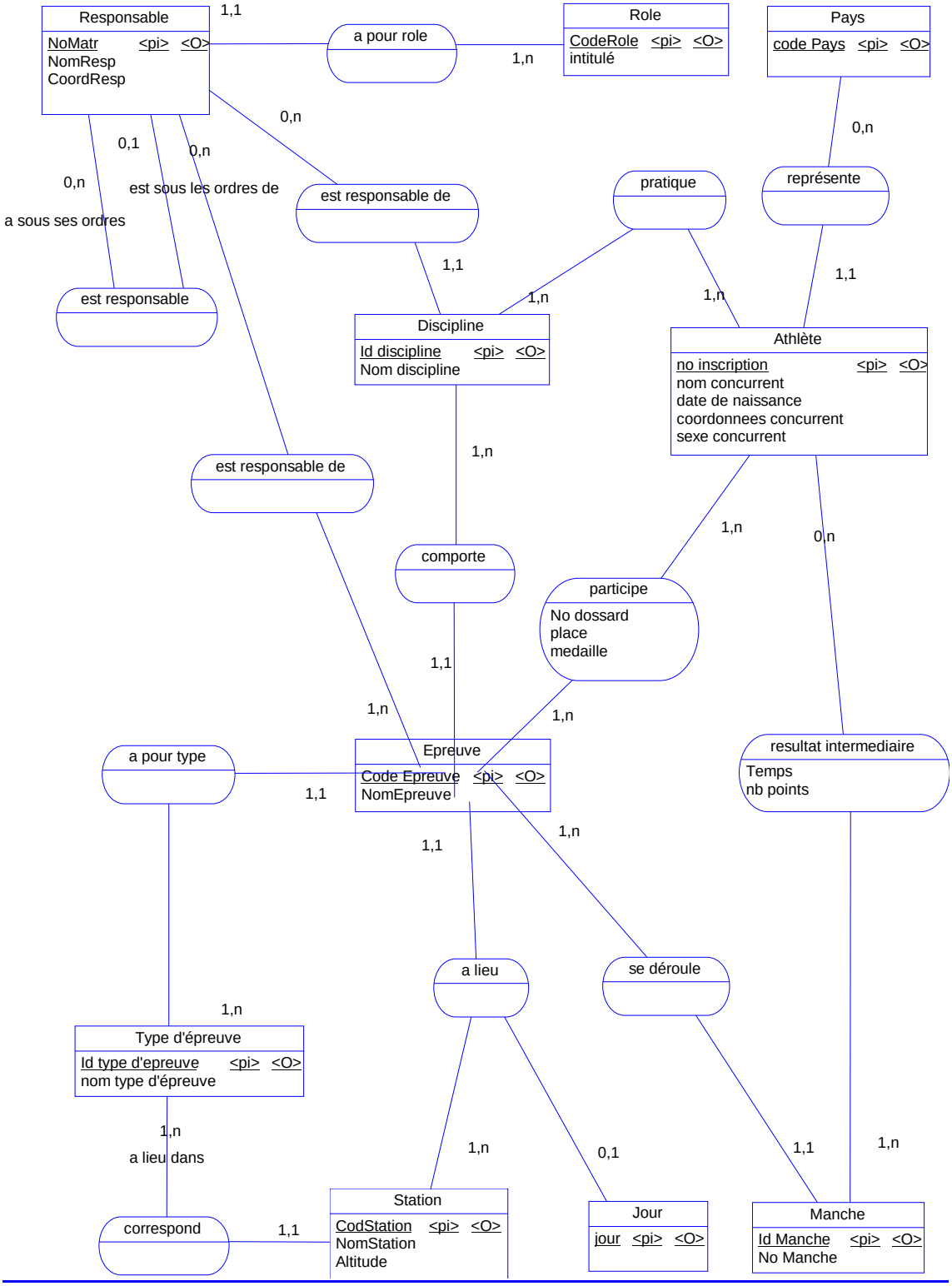
Chaque concurrent représente un Pays et participe à une ou plusieurs épreuves. Il peut pratiquer plusieurs disciplines. Dans les manches d'une épreuve il porte toujours le même numéro de dossard mais en change pour une autre épreuve.

A l'issue de chaque manche , le système doit enregistrer , selon l'épreuve , le temps (Ski , ….) ou le nombre de points de chaque concurrent. A la fin de l'épreuve , il mémorisera la place et éventuellement la médaille (OR , ARGENT , BRONZE) de chacun des athlètes.

On distinguera les concurrents hommes des femmes car les épreuves ne sont pas mixte. Ils sont identifiés par un numéro d'inscription aux jeux et ont un Nom , une date de naissance et des coordonnées.

Tous les pays sont codifiés (FR , GB , USA ….).

1-MODELE CONCEPTUEL DE DONNEES



2-MODELE LOGIQUE DE DONNEES

RESPONSABLE (NoMatr, NomResp, CoordResp, *CodRole, NoMatrSup*)
ROLE (CodRole, intitulé)
DISCIPLINE (Id discipline, Nom discipline, *NoMatrResp*)
PRATIQUE (no inscription, Id discipline)
RESPONSABLE_EPREUVE (NoMatr, Code épreuve)
EPREUVE (Code épreuve, NomEpreuve, *Id discipline, Id type d'épreuve, Nom station*, jour)
TYPE_EPREUVE (Id type d'épreuve, NomTypeEpreuve)
STATION (CodStation, Nom station, altitude, *Id type d'épreuve*)
MANCHE (Id manche, Nom manche, *Code épreuve*)
PARTICIPE (Code épreuve, No inscription, no dossard, place, médaille)
ATHLETE (No inscription, nom concurrent, date de naissance, coordonnées concurrent, sexe concurrent, *Cod pays*)
PAYS (Codpays, NomPays)
RESULTAT_INTERMEDIAIRE (No inscription, Id manche, Temps, nb points)

3-MODELE LOGIQUE DE DONNEES-TABLEAU DES CONTRAINTES

Table RESPONSABLE

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
NoMatr	Numérique	3	oui	oui					
NomResp	Caractère	20		oui					
CoordResp	Caractère	60		oui					
NoMatrSup	Numérique	3		oui					RESPONSABLE
CodeRole	Caractère	20		oui					ROLE

Table ROLE

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
CodeRole	Numerique		oui	oui	+1				
intitulé Rôle	Caractère	20	oui	oui					

Table DISCIPLINE

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
Id discipline	Numérique	3	oui	oui					
Nom discipline	Caractère	20		oui					
NoMatrResp	Numérique	3		oui					RESPONSABLE

Table PRATIQUE

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
No inscription	Numérique	3	oui	oui					ATHLETE
Id discipline	Numérique	3	oui	oui					DISCIPLINE

Table RESPONSABLE_EPREUVE

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
NoMatr	Numérique	3	oui	oui					RESPONSABLE
Code epreuve	Numérique	3	oui	oui					EPREUVE

Table EPREUVE

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
Code épreuve	Numérique	3	oui	oui					
Nom Epreuve	Caractère	20		oui					
Id discipline	Numérique	3		oui					DISCIPLINE
Id type d'épreuve	Numérique	3		oui					TYPE EPREUVE
Nom station	Caractère	20		oui					STATION
Jour	Date	JJ/MM/AA		oui					

Table TYPE_EPREUVE

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
Id type d'épreuve	Numérique	3	oui	oui					
Nom type d'épreuve	Caractère	20		oui					

Table STATION

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
N°Station	Numérique	5	oui	oui	+1				
Nom station	Caractère	20	oui	oui					
Altitude	Numérique	5		non					
Id type d'épreuve	Numérique	3	oui	oui					TYPE_EPREUVE

Table MANCHE

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
Id manche	Numérique	3	oui	oui					
Nom manche	Caractère	20		oui					
Code épreuve	Numérique	3		oui					EPREUVE

Table PARTICIPE

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
Code épreuve	Numérique	3	oui	oui					EPREUVE
No inscription	Numérique	3	oui	oui					ATHLETE
No dossard	Numérique	4		oui					
place	Numérique	3		non					
Médaille	Caractère	10		non		Or, Argent, Bronze			

Table ATHLETE

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
No inscription	Numérique	3	oui	oui					
Nom concurrent	Caractère	20		oui					
Date de naissance	Date	JJ/MM/AA		oui					
coordonnées concurrent	Caractère	60		oui					
sexe concurrent	Caractère	1		oui		M,F			
code pays	Caractère	3		oui					

Table PAYS

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
Code pays	Caractère	3	oui	oui					
Nom pays	Caractère	20		oui					

Table RESULTAT_INTERMEDIAIRE

Contraintes / Attributs	DOMAINE						INTER		
	Type	Structure	Unique	Obligatoire	Evolutivité	Valeur	Propriété	N-Uplets	Relation
No inscription	Numérique	3	oui	oui					ATHLETE
Id manche	Numérique	3	oui	oui					MANCHE
Temps	Numérique	HH:MM:SS		non					
nb points	Numérique	3		non					

4-CREATION DE LA BASE D E DONNEES

```
DROP TABLE ResInter;
DROP TABLE Participe;
DROP TABLE Pratique;
DROP TABLE RespEpreuve;
DROP TABLE Manche;
DROP TABLE Epreuve;
DROP TABLE Discipline;
DROP TABLE Responsable;
DROP TABLE Athlete;
DROP TABLE Role;
DROP TABLE Pays;
DROP TABLE Station;
DROP TABLE TypeEpreuve;

CREATE TABLE Role
	(
CodRole smallint IDENTITY(1,1) not null,
	IntitRole varchar(30) not null,
	Constraint pk_role
		primary key (CodRole) );


CREATE TABLE Pays
	(
CodPays varchar(3) not null,
	NomPays varchar(20) not null,
	Constraint pk_pays
		primary key (CodPays) );

CREATE TABLE TypeEpreuve
	(
	IDTypeEpr int IDENTITY(1,1) not null,
	NomTypeEpr varchar(20) not null,
	Constraint pk_typeepr
		primary key (IDTypeEpr) );

CREATE TABLE Station
	(
	CodStation smallint IDENTITY(1,1) not null,
	NomStation varchar(20) not null,
	Alti int not null,
	IDTypeEpr int not null,
	Constraint pk_station
		primary key (CodStation),
	Constraint fk_TypeEpreuve
		foreign key (IDTypeEpr) references TypeEpreuve(IDTypeEpr));


CREATE TABLE Responsable
	(
	NoMatrResp int not null,
	NomResp varchar(20) not null,
	CoordResp varchar(60) not null,
	CodeRole smallint not null,
	NoMatrSup int,
	Constraint pk_resp
		primary key (NoMatrResp),
```

```
        Constraint fk_respRole
                foreign key (CodeRole) references Role(CodRole),
        Constraint fk_supRespSup
                foreign key (NoMatrSup) references Responsable(NoMatrResp),
        Constraint ck_diff
                check (NoMatrResp<>NoMatrSup ));


CREATE TABLE Discipline
        (
IDDisc int IDENTITY(1,1) not null,
        NomDisc varchar(20) not null,
        NoMatrResp int not null,
        Constraint pk_discipline
                primary key (IDDisc),
        Constraint fk_respDisc
                foreign key (NoMatrResp) references Responsable(NoMatrResp));

CREATE TABLE Athlete
        (
NoInscrit int IDENTITY(1,1) not null,
        NomAthl varchar(30) not null,
        DateNaiss datetime not  null,
        CoordAthl varchar(60) not null,
        SexeAthl char not null,
        CodPays varchar(3) not null,
        Constraint pk_athlete
                primary key (noInscrit),
        Constraint fk_athletePays
                foreign key (CodPays) references Pays(CodPays),
        Constraint ck_sexe
                check (SexeAthl='M' or SexeAthl='F'));


CREATE TABLE Epreuve
        (
CodEpr int IDENTITY(1,1) not null,
        NomEpr varchar(20) not null,
        IdDisc int not null,
        IdType int not null,
        CodStation smallint not null,
        Jour datetime not null,
        Constraint pk_epreuve
                primary key (CodEpr),
        Constraint fk_epreuveDisc
                foreign key (IdDisc) references Discipline(IDDisc),
        Constraint fk_epreuveType
                foreign key (IdType) references TypeEpreuve(IDTypeEpr),
        Constraint fk_epreuveStation
                foreign key (CodStation) references Station(Cod Station));

CREATE TABLE Manche
        (
IdManche int not null,
        NomManche varchar(20) not null,
        CodEpr int not null,
        Constraint pk_manche
                primary key (IdManche),
        Constraint fk_mancheEpr
                foreign key (CodEpr) references Epreuve(CodEpr),);
```

```
CREATE TABLE RespEpreuve
	(
NoMatrResp int not null,
	CodEpr int not null,
	Constraint pk_respEpr
		primary key (NoMatrResp,CodEpr),
	Constraint fk_respEprResp
		foreign key (NoMatrResp) references Responsable(NoMatrResp),
	Constraint fk_respRespEpr
		foreign key (CodEpr) references Epreuve(CodEpr));

CREATE TABLE Pratique
	(
IdDisc int not null,
	NoInscrit int not null,
	Constraint pk_pratique
		primary key (IdDisc,NoInscrit),
	Constraint fk_pratAthlete
		foreign key (NoInscrit) references Athlete(NoInscrit),
	Constraint fk_pratDisc
		foreign key (IdDisc) references Discipline(IdDisc));

CREATE TABLE Participe
	(
NoInscrit int not null,
	CodEpr int not null,
	NoDossard int not null,
	Place int ,
	Medaille varchar(10),
	Constraint pk_participe
		primary key (CodEpr,NoInscrit),
	Constraint fk_partAthlete
		foreign key (NoInscrit) references Athlete(NoInscrit),
	Constraint fk_partEpreuve
		foreign key (CodEpr) references Epreuve(CodEpr),
	Constraint ck_medaille
		check (Medaille='Or' or Medaille='Argent' or Medaille='Bronze' ));

CREATE TABLE ResInter
	(
NoInscrit int not null,
	IdManche int not null,
	Temps datetime ,
	NbPts int,
	Constraint pk_resInter
		primary key (NoInscrit,IdManche),
	Constraint fk_resInterAthlete
		foreign key (NoInscrit) references Athlete(NoInscrit),
	Constraint fk_resInterManche
		foreign key (IdManche) references Manche(IdManche));
```

5-JEU D'ESSAI

```
DELETE from ResInter;
DELETE from Participe;
DELETE from Pratique;
DELETE from RespEpreuve;
DELETE from Manche;
DELETE from Epreuve;
DELETE from Athlete;
DELETE from Discipline;
DELETE from Responsable;
DELETE from Station;
DELETE from Pays;
DELETE from Role;
DELETE from TypeEpreuve;

INSERT INTO Role   (IntitRole)
        values ('responsable general');
INSERT INTO Role (IntitRole)
        values ('responsable discipline ');
INSERT INTO Role (IntitRole)
        values ('responsable epreuve');
INSERT INTO Role (IntitRole)
        values ('juge a l''arrivee');
INSERT INTO Role (IntitRole)
        values ('juge au départ');
INSERT INTO Role (IntitRole)
        values ('chronometreur');
INSERT INTO Role  (IntitRole)
        values ('juge 1');
INSERT INTO Role (IntitRole)
        values ('juge 2');
INSERT INTO Role (IntitRole)
        values ('juge 3');
INSERT INTO Role (IntitRole)
        values ('juge 4');

INSERT INTO Responsable (NoMatrResp, NomResp,CoordResp,CodeRole)
        values (1, 'Dupont', 'France',1);
INSERT INTO Responsable (NoMatrResp,NomResp,CoordResp,CodeRole,NoMatrSup)
        values (2,'Tardy', 'France', 2,1);
INSERT INTO Responsable (NoMatrResp,NomResp,CoordResp,CodeRole,NoMatrSup)
        values (3,'Tronet', 'France', 2,1);
INSERT INTO Responsable (NoMatrResp,NomResp,CoordResp,CodeRole,NoMatrSup)
        values (4,'Durand', 'France', 3,2);
INSERT INTO Responsable (NoMatrResp,NomResp,CoordResp,CodeRole,NoMatrSup)
        values (5,'Martin', 'Suisse', 3,3);
INSERT INTO Responsable (NoMatrResp,NomResp,Co ordResp,CodeRole,NoMatrSup)
        values (6,'Dujardin', 'France', 4,4);
INSERT INTO Responsable (NoMatrResp,NomResp,CoordResp,CodeRole,NoMatrSup)
        values (7,'Clemart', 'France',5,4);
INSERT INTO Responsable (NoMatrResp,NomResp,CoordResp,CodeRole,NoMatrSup)
        values (8,'Smith', 'Grande-Bretagne', 6,4);
INSERT INTO Responsable (NoMatrResp,NomResp,CoordResp,CodeRole,NoMatrSup)
        values (9,'Wesson', 'Grande-Bretagne', 7,5);
INSERT INTO Responsable (NoMatrResp,NomResp,CoordResp,CodeRole,NoMatrSup)
        values (10,'Toinat', 'Luxembourg', 8,5);
INSERT INTO Responsable (NoMatrResp,NomResp,CoordResp,CodeRole,NoMatrSup)
        values (11,'Alvarez', 'Mexique', 9,5);
INSERT INTO Responsable (NoMatrResp,NomResp,CoordResp,CodeRole,NoMatrSup)
```

```
        values (12,'Riegel', 'Allemagne', 10,5);


INSERT INTO Pays
        values ('GB', 'Grande-Bretagne');
INSERT INTO Pays
        values ('FRA', 'France');
INSERT INTO Pays
        values ('SUI', 'Suisse');
INSERT INTO Pays
        values ('ALL', 'Allemagne');
INSERT INTO Pays
        values ('CAN', 'Canada');
INSERT INTO Pays
        values ('RUS', 'Russie');
INSERT INTO Pays
        values ('NOR', 'Norvege');

INSERT INTO TypeEpreuve (NomTypeEpr)
        values ('Ski');
INSERT INTO TypeEpreuve (NomTypeEpr)
        values ('Patinage');

INSERT INTO Station (NomStation, Alti,IDTypeEpr)
        values ('Chamonix',1500, 1);
INSERT INTO Station (NomStation, Alti,IDTypeEpr)
        values ('Tignes',1500, 2);

INSERT INTO Discipline (NomDisc,NoMatrResp)
        values ('Ski alpin',2);
INSERT INTO Discipline (NomDisc,NoMatrResp)
        values ('Patinage artistique',3);

INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Dubois Pierre','05/02/1971','Dinan','M','FRA');
INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Carpenter John','14/05/1972','Londres','M','GB');
INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Weiss Charles','31/03/1971','Geneve','M','SUI');

INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Fichtner Andrea','14/03/1975','Berlin','F','ALL');
INSERT INTO Athlete (NomAthl ,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Desbois Marie','01/05/1977','Grenoble','F','FRA');
INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Kowtun Regis','06/06/1972','Moscou','M','RUS');
INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Karamazov Tatiana','26/05/1985','St-Petersbourg','F','RUS');
INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Lambert Denis','21/03/1976','Montreal','M','CAN');
INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Wilson Julia','03/12/1980','Ottawa','F','CAN');
INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Trevor Audrey','21/07/1981','Birmingham','F','GB');
INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Godisch Martin','30/04/1982','Hambourg','M','ALL');
INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Olafsen Lars','16/09/1979','Oslo','M','NOR');
INSERT INTO Athlete (NomAthl,DateNaiss,CoordAthl,SexeAthl,CodPays)
        values ('Erikson Tania','14/08/1978','Oslo','F','NOR');
```

```
INSERT INTO Epreuve (NomEpr,IdDisc,IdType,CodStation,Jour)
        values ('Slalom Geant Homme',1,1,1,'09/02/2002');
INSERT INTO Epreuve (NomEpr,IdDisc,IdType,CodSta tion,Jour)
        values ('Slalom Geant Femme',1,1,1,'11/02/2002');
INSERT INTO Epreuve (NomEpr,IdDisc,IdType,CodStation,Jour)
        values ('Patinage Femme',2,2,2,'10/02/2002');
INSERT INTO Epreuve (NomEpr,IdDisc,IdType,CodStation,Jour)
        values ('Patinage Homme',2,2 ,2,'12/02/2002');

INSERT INTO Manche (IdManche,NomManche,CodEpr)
        values (1,'Slalom Homme',1);
INSERT INTO Manche (IdManche,NomManche,CodEpr)
        values (2,'Slalom Femme',2);
INSERT INTO Manche (IdManche,NomManche,CodEpr)
        values (3,'Libre Femme',3);
INSERT INTO Manche (IdManche,NomManche,CodEpr)
        values (4,'Imposé Femme',3);
INSERT INTO Manche (IdManche,NomManche,CodEpr)
        values (5,'Libre Homme',4);
INSERT INTO Manche (IdManche,NomManche,CodEpr)
        values (6,'Imposé Homme',4);

INSERT INTO RespEpreuve
        values (4,1);
INSERT INTO RespEpreuve
        values (4,2);
INSERT INTO RespEpreuve
        values (5,3);
INSERT INTO RespEpreuve
        values (5,4);

-- athlete pratique discipline
INSERT INTO Pratique
        values (2,1);
INSERT INTO Pratique
        values (1,2);
INSERT INTO Pratique
        values (1,3);
INSERT INTO Pratique
        values (1,4);
INSERT INTO Pratique
        values (2,5);
INSERT INTO Pratique
        values (2,6);
INSERT INTO Pratique
        values (2,7);
INSERT INTO Pratique
        values (2,8);
INSERT INTO Pratique
        values (2,9);
INSERT INTO Pratique
        values (1,10);
INSERT INTO Pratique
        values (1,11);
INSERT INTO Pratique
        values (1,12);
INSERT INTO Pratique
        values (1,13);
```

```
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (1,4,1);
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (6,4,2);
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (8,4,3);
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (2,1,1);
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (3,1,2);
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (11,1,3);
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (12,1,4);
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (5,3,1);
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (7,3,2);
INSERT INTO Participe (NoInscrit, CodEpr,NoDossard)
        values (9,3,3);
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (4,2,1);
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (10,2,2);
INSERT INTO Participe (NoInscrit,CodEpr,NoDossard)
        values (13,2,3);

INSERT INTO Re sInter (NoInscrit,IdManche)
        values (2,1);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (3,1);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (11,1);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (12,1);
INSERT INTO ResInter (NoInscrit,IdManche )
        values (4,2);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (10,2);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (13,2);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (5,3);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (7,3);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (9,3);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (5,4);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (7,4);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (9,4);
INSERT INTO ResInter (NoInscrit,IdMan che)
        values (1,5);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (6,5);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (8,5);
INSERT INTO ResInter (NoInscrit,IdManche)
```

```
        values (1,6);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (6,6);
INSERT INTO ResInter (NoInscrit,IdManche)
        values (8,6);
```

6-MISE A JOUR DES TABL ES DE JO AVEC LES RESULTATS

```
-- patinage femme
UPDATE ResInter
        SET NbPts=5.88
        WHERE (NoInscrit=5 and IdManche=3);
UPDATE ResInter
        SET NbPts=5.21
        WHERE (NoInscrit=7 and IdManche=3) ;
UPDATE ResInter
        SET NbPts=4.78
        WHERE (NoInscrit=9 and IdManche=3);
UPDATE ResInter
        SET NbPts=5.61
        WHERE (NoInscrit=5 and IdManche=4);
UPDATE ResInter
        SET NbPts=5.95
        WHERE (NoInscrit=7 and IdManche=4);
UPDATE ResInter
        SET NbPts=5.30
        WHERE (NoInscr it=9 and IdManche=4);

UPDATE Participe
        SET     Place=1,
                Medaille='Or'
        WHERE NoInscrit=5;
UPDATE Participe
        SET     Place=2,
                Medaille='Argent'
        WHERE NoInscrit=7;
UPDATE Participe
        SET     Place=3,
                Medaille='Bronze'
        WHERE NoInscrit=9;

-- patinage homme
UPDATE ResInter
        SET NbPts=5.75
        WHERE (NoInscrit=1 and IdManche=5);
UPDATE ResInter
        SET NbPts=5.84
        WHERE (NoInscrit=6 and IdManche=5);
UPDATE ResInter
        SET NbPts=5.03
        WHERE (NoInscrit=8 and IdManche=5);
UPDATE ResInter
        SET NbPts=4.56
        WHERE (NoInscrit=1  and IdManche=6);
UPDATE ResInter
        SET NbPts=5.32
        WHERE (NoInscrit=6 and IdManche=6);
UPDATE ResInter
        SET NbPts=5.21
        WHERE (NoInscrit=8 and IdManche=6);

UPDATE Participe
        SET     Place=2,
                Medaille='Argent'
        WHERE NoInscrit=1;
UPDATE Participe
```

```
        SET     Place=1,
                Medaille='Or'
        WHERE NoInscrit=6;
UPDATE Participe
        SET     Place=3,
                Medaille='Bronze'
        WHERE NoInscrit=8;

-- slalom homme
UPDATE ResInter
        SET Temps='00:04:23'
        WHERE (NoInscrit=2 and IdManche=1);
UPDATE ResInter
        SET Temps='00:05:02'
        WHERE (NoInscrit=3 and IdManche=1);
UPDATE ResInter
        SET Temps='00:04:51'
        WHERE (NoInscrit=11 and IdManche=1);
UPDATE ResInter
        SET Temps='00:04:44'
        WHERE (NoInscrit=12 and IdManche=1);


UPDATE Participe
        SET     Place=2,
                Medaille='Argent'
        WHERE NoInscrit=12;
UPDATE Participe
        SET     Place=1,
                Medaille='Or'
        WHERE NoInscrit=2;
UPDATE Participe
        SET     Place=3,
                Medaille='Bronze'
        WHERE NoInscrit=11;
UPDATE Participe
        SET     Place=4
        WHERE NoInscrit=3;

--slalom femme
UPDATE ResInter
        SET Temps='00:05:12'
        WHERE (NoInscrit=4 and IdManche=2);
UPDATE ResInter
        SET Temps='00:04:59'
        WHERE (NoInscrit=10 and IdManche=2);
UPDATE ResInter
        SET Temps='00:05:24'
        WHERE (NoInscrit=13 and IdManche=2);

UPDATE Participe
        SET     Place=2,
                Medaille='Argent'
        WHERE NoInscrit=4;
UPDATE Participe
        SET     Place=1,
                Medaille='Or'
        WHERE NoInscrit=10;
UPDATE Participe
        SET     Place=3,
```

```
            Medaille='Bronze'
WHERE NoInscrit=13;
```

7-REQUETES

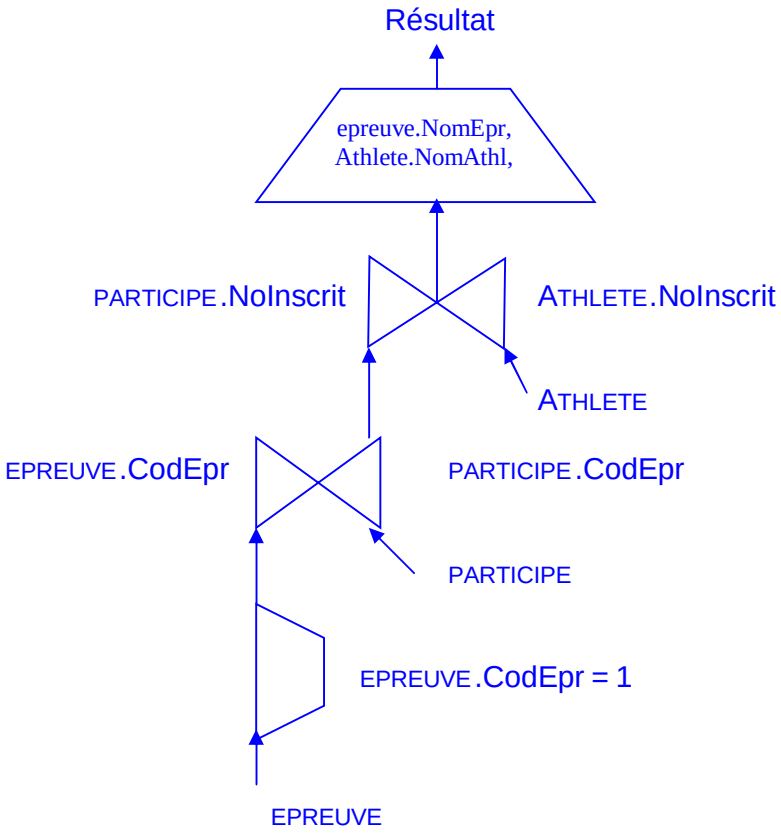
REQ1 : RESULTATS DU SLALOM GEANT HOMME :

ALGEBRE RELATIONNELLE :

Analytique :

R1=REST (EPREUVE ; CodEpr = 1)
R2=JOIN (EPREUVE , PARTICIPE , ATHLETE; (EPREUVE .CodEpr = PARTICIPE .CodEpr) ET (PARTICIPE .NoInscrit = ATHLETE .NoInscrit)
R3=PROJ(R2, EPREUVE .NomEpr, ATHLETE .NomAthl, PARTICIPE .medaille)

Graphique :



REQUETE SQL :

```
Select epreuve.NomEpr, Athlete.NomAthl, participe.medaille
from epreuve inner join (participe inner join Athlete
on participe.NoInscrit = Athlete.NoInscrit)
on epreuve.CodEpr = participe.CodEpr
where epreuve.CodEpr = 1
order by participe.place
```

Support de Formation

Bases de données :

Présentation générale et Méthode de conception

Lécu Régis : CFPA Champs/Marne, Formation APIAI (origine CFPA Dijon)

CHAPITRE GENERALITES SUR LES BASES DE DONNEES

Définition et Historique

- Une base de données est un ensemble structuré de données enregistrées sur des supports informatisés, pouvant satisfaire simultanément plusieurs utilisateurs de façon sélective, en un délai raisonnable.
- Le concept de Base de Données (BDD) est apparu vers 1960, face au nombre croissant d'informations que les entreprises devaient gérer et partager :

- chaque nouvelle application créait alors ses propres fichiers de données et ses propres programmes ;
- le concept de base de données va à l'encontre de cette façon de procéder : il permet la centralisation, la coordination, l'intégration et la diffusion de l'information archivée.

- La base de données enregistre les faits ou événements qui surviennent dans la vie d'un organisme, pour les restituer à la demande : elle permet également de tirer des conclusions en rapprochant plusieurs faits élémentaires.
- Les données peuvent être manipulées par plusieurs utilisateurs ayant des **vues différentes** sur ces données ("points de vue" différents).
- La structure d'ensemble des données suit une définition rigoureuse appelée **SCHEMA**.

Facteurs liés au développement des SGBD :

- augmentation des capacités mémoire, et diminution des temps d'accès
- apparition sur le marché d'applications fiables et diversifiées, qui doivent partager leurs données
- développement des systèmes de gestion en temps réel : "Gestion transactionnelle"
- approche globale "orientée données" des problèmes de gestion : les données sont organisées de façon rationnelle plutôt que définies au coup par coup selon les applications à réaliser.

Rappel sur les systèmes de gestion de fichiers

Toute manipulation de fichier exige trois niveaux d'intervention, et trois couches logicielles :

- Gestion du support physique : disques durs, disquette, streamers…

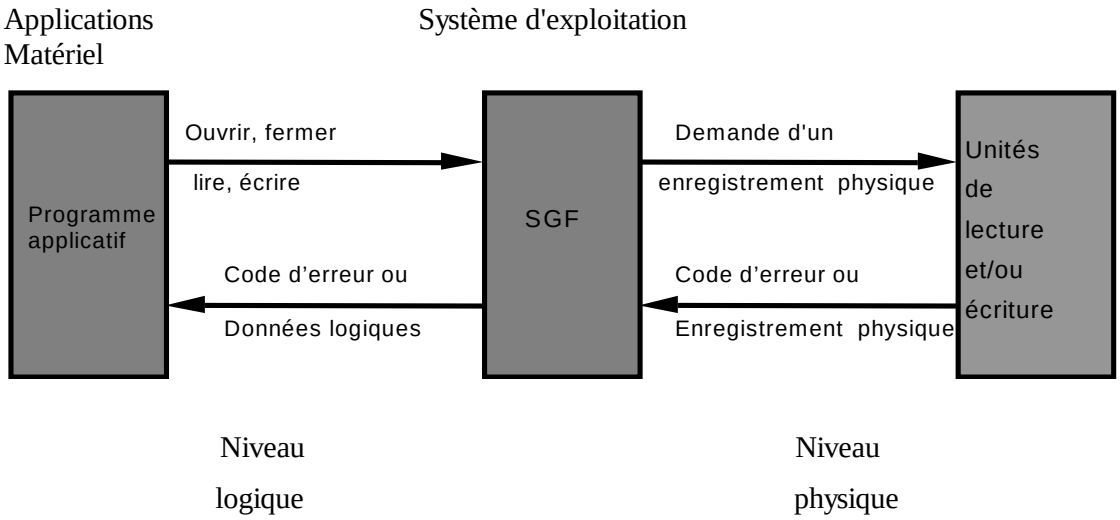
ð *Pilote d'entrées-sorties (Driver)*

- Gestion des structures internes des fichiers, et des méthodes d'accès : ouverture, fermeture, lecture, écriture…

ð *Système de gestion de fichiers (SGF)*

- Gestion des contenus : calculs, tests, affichages ...

ð *Programmes applicatifs*

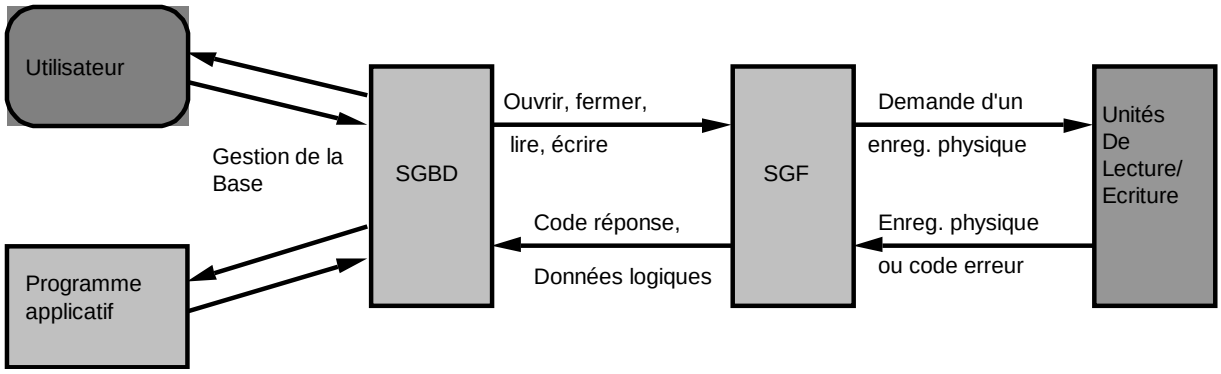


Système de Gestion de Base de données : SGBD

- Ensemble des programmes et des langages de commande qui permettent de :

- définir des "bases de données", et des relations entre les éléments de chaque base ;
- spécifier le traitement de ces données : interrogations, mises à jour, calculs, extractions...

- Le SGBD reçoit des commandes aussi bien des programmes d'application que des utilisateurs : il commande les manipulations de données, généralement par l'intermédiaire d'un SGF.



CHAPITRE OBJECTIFS DE L'APPROCHE SGBD

- Pour pallier aux inconvénients des méthodes classiques de gestion de fichiers, les SGBD visent quatre objectifs : intégration et corrélation, flexibilité (indépendance), disponibilité, sécurité.
- Ces objectifs exigent une distinction nette entre les données et les procédures de manipulation de ces données : aux données, on associera une fonction **d'administration des données**, aux procédures de manipulation une **fonction de programmation**.

Intégration et corrélation

Dans les systèmes classiques, chaque application gère ses données dans ses propres "fichiers", d'où :

- Un risque de redondance, et un danger d'incohérence des données
  - La même donnée peut appartenir à plusieurs applications, induisant une déperdition de stockage.
  - Toute modification de cette donnée est à enregistrer plusieurs fois : si cette mise à jour multiple n'est pas effectuée correctement, les données deviennent incohérentes.
  - Le coût de la mise à jour augmente du fait de la multiplication des entrées -sorties physiques.

- Une difficulté pour créer de nouveaux traitements
  - Les nouvelles applications entraînent des duplications supplémentaires d e données.
  - Leur intégration avec les applicatifs en exploitation entraîne des modifications importantes.

Dans l'approche SGBD, un "réservoir" commun (**intégration**) est constitué, représentant une modélisation (**corrélation**) aussi fidèle que possible de l'organisation réelle de l'entreprise :

- Toutes les applications puisent dans ce réservoir, les données qui les concernent, évitant ainsi les duplications.
- Mais le partage des données entre les utilisateurs pose le problème de la synchronisation des accès concurrents.

ag Flexibilit é ou indépendance

- Dans les systèmes classiques, tout changement intervenant dans le stockage des données (support, méthode d'accès physique) entraîne des modifications lourdes des applications correspondantes.
- L'approche SGBD poursuit trois objectifs, pour assurer l'indépendance des données par rapport aux traitements :

- indépendance physique: tout changement de support, de méthode d'accès reste transparent au niveau de l'utilisateur.
- indépendance logique : les programmes d'applic ation sont rendus transparents à une modification dans l'organisation logique globale, par la définition de sous -schémas couvrant les besoins spécifiques en données.
- indépendance vis-à-vis des stratégies d'accès : l'utilisateur n'a plus à prendre en charge l'écriture des procédures d'accès aux données. Il n'a donc pas à intégrer les modifications tendant à optimiser les chemins d'accès (ex: création d'index).

2.3 Disponibilit é

- Le choix d'une approche SGBD ne doit pas se traduire par des temps de traiteme nt plus longs que ceux des systèmes antérieurs.
- L'utilisateur doit ignorer l'existence d'utilisateurs concurrents.
- L'aspect "performance" est donc crucial dans la mise en oeuvre d'une base de données. Un tel objectif ne peut être atteint que si la concepti on d'une base de données est menée de façon rigoureuse avec un découpage fonctionnel adéquat. Les règles et contraintes inhérentes sont évoquées lors de l'apprentissage d'une méthodologie d'analyse (exemple MERISE).

2.4 Sécurit é

- La sécurité des données recouvre deux aspects :

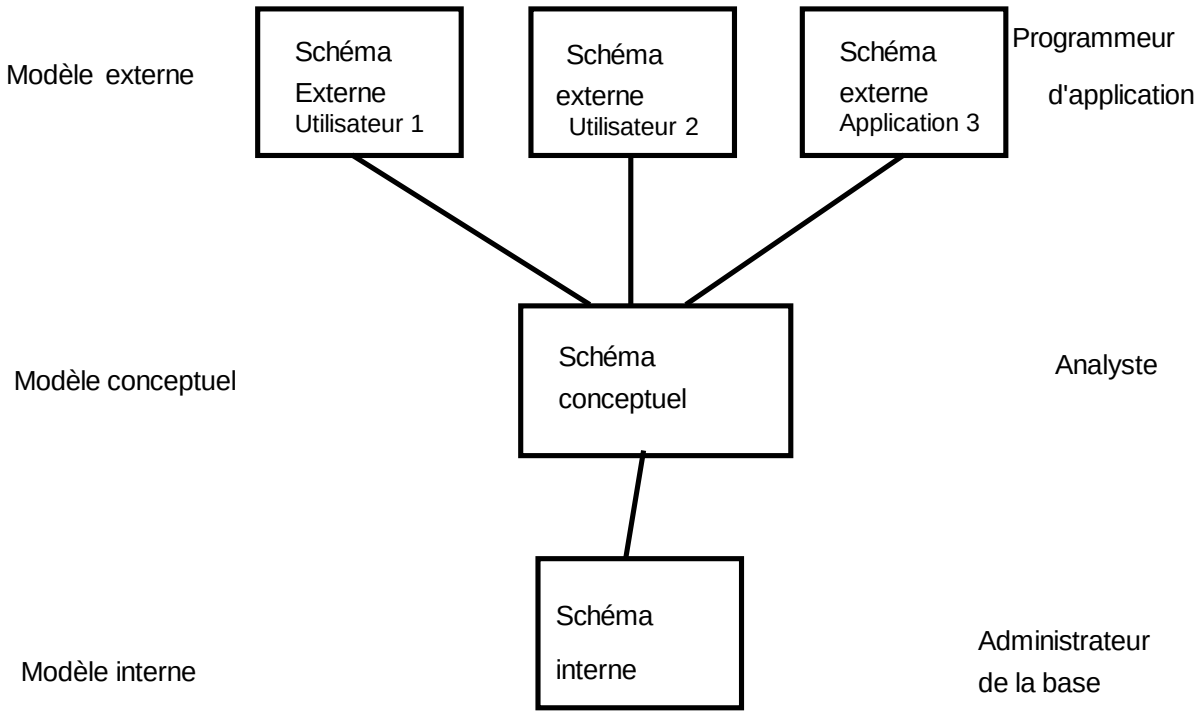
- **l'intégrité**, ou protection contre l'accès invalide (erreurs ou pannes), et contre l'incohérence des données vis-à-vis des contraintes de l'entreprise.
- **la confidentialité**, ou protection contre l'accès non autorisé ou la modification i llégale des données.

- Pour ne pas trop affecter les performances, la sécurité doit également être prise en compte dès la phase de conception.

CHAPITRE ARCHITECTURE FONCTIONNELLE D'UN SGBD : ANSI-SPARC

- Dans le cadre du groupe de normalisation nord américain (ANSI), un groupe d'études a été créé en 69, *Standard Planning and Requirement Committee* (SPARC) avec pour mission, une standardisation des SGBD.

- Les travaux ont abouti en 75 (ANSI 75) par la proposition d'une architecture multi-niveaux : à chaque niveau fonctionnel, sont associés un modèle et un schéma de données, un langage de description de données (LDD) permettant de décrire les données du schéma, et un langage de manipulation de données (LMD) permettant de les utiliser (accès pour consultation, mise à jour...).



Niveau conceptuel

- C'est une abstraction aussi fidèle que possible, de l'univers de l'entreprise, après modélisation et indépendamment de toute référence à l'utilisation et à l'implantation en machine.
- Le modèle conceptuel de données (MCD) permet le passage d'un concret inaccessible (l'univers réel) à un abstrait manipulable : le schéma conceptuel. Celui-ci peut donc être considéré comme la description du contenu de la base : c'est le résultat d'un travail d'analyse et de conception d'un système d'information automatisé.

- Un schéma conceptuel doit offrir les caractéristiques suivantes :

- puissance de représentation : aspects structurels, contraintes existant dans l'univers réel.
- stabilité et flexibilité : l'ajout d'une nouvelle donnée ou d'une nouvelle contrainte ne doit pas entraîner de changement important dans le schéma.
- simplicité de compréhension : nombre d'éléments réduit, dissociation claire des différents concepts.
- simplicité d'utilisation : nombre restreint d'outils ou de primitives de manipulation.
- base formelle : la définition du schéma doit s'appuyer sur une méthode rigoureuse, mathématique, pour éviter toute ambiguïté d'interprétation et pour garantir la fiabilité des données.

- Pour aboutir au schéma conceptuel, l'analyste doit repérer dans le réel, et recenser de manière exhaustive, toutes les entités et toutes les associations :

- Une **entité** peut être définie comme une personne, un objet, un lieu, un statut, un événement qui ont une existence dans le monde réel. C'est un objet concret ou abstrait, possédant un certain nombre de caractéristiques spécifiques (exemple : le produit x coûte y francs).
- Généralement, les entités du monde réel se manifestent à travers des faits élémentaires.
- Certains faits faisant intervenir plusieurs entités, il apparaît la notion d'**association**. Une association (ou **lien**) est un ensemble de deux ou plusieurs entités, chacune d'elles jouant un rôle particulier.

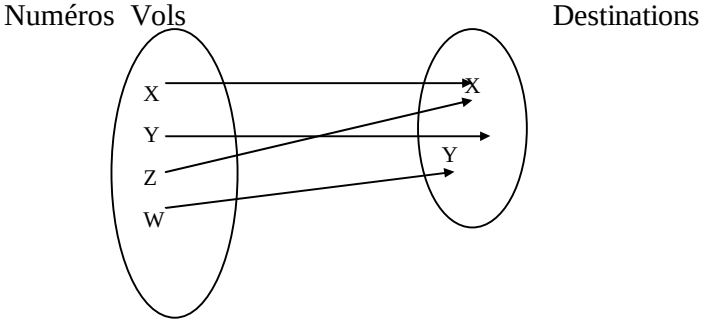
Exemple : le fait que la "voiture x" appartienne à la "personne y" est u ne association entre les entités "voiture " et "personne".

- Selon la notation CODASYL, trois types de liens peuvent être envisagés :

- <u>les liens fonctionnels notés N : 1</u>

On a un lien fonctionnel N:1 de A vers B si toute occurrence de A détermine au plus une occurrence de B, et si à toute occurrence de B, correspond un nombre quelconque d'occurrences de A.

Exemple : dans une compagnie aérienne, connaissant le numéro d'un vol, on en déduit d'une manière unique la destination, mais plusieurs vols peuvent avoir la même destination.



- les liens hiérarchiques notés 1 : N.

On a un lien hiérarchique 1:N de A vers B si une occurrence de A peut déterminer un nombre quelconque d'occurrences de B et si, à une occurrence de B, correspond au plus une occurrence de A.

Exemple : la polygamie est un lien 1 : N de "homme" vers "femme".

- les liens maillés notés N : M.

On a un lien maillé de A vers B s'il n'existe aucune restriction sur le nombre d'occurrences de A et B intervenant dans le lien.

Exemple : dans un lycée donné, un enseignant peut dispenser des cours dans plusieurs matières différentes ; de la même façon, une matière peut être dispensée par plusieurs enseignants.

Niveau externe

- Le niveau externe comprend les "vues" spécifiques définies pour la manipulation des données. Il prend en compte les contraintes d'accès imposées par la nature des applications à considérer (indépendamment des caractéristiques techniques) et exprime les besoins en données des différents utilisateurs, ou applications.

- Le modèle logique des données (MLD) utilisé à ce niveau externe peut différer de celui utilisé au niveau conceptuel. Ainsi, certaines vues peuvent ne pas être construites dans la base, mais déduites par calcul à partir de certaines données du schéma conceptuel (exemple : ancienneté obtenue par différence entre année en cours et annnée d'embauche dans la société).

Niveau interne ou Physique

- Il correspond à la représentation en machine, aussi efficace que possible, du schéma conceptuel : le schéma physique intègre les caractéristiques techniques (choix du SGBD, du matériel, du système d'exploitation…).

- L'efficacité doit tenir compte d'une part des contraintes d'implantation (taille des disques, optimisation du système de fichiers…), d'autre part des critères d'utilisation (traitement interactif ou en batch, selon la fréquence d'utilisation et la durée du traitement…).

CHAPITRE FONCTIONNEMENT D'UN SGBD

0g Chronologie des opérations dans l'interrogation d'un SGDB

Un programme d'application A émet une demande de lecture de données au SGBD sur une des bases :

- Le SGBD traite la demande en consultant le sous-schéma externe relatif au programme d'application A, obtenant ainsi la description des données.
- Le SGBD consulte le schéma conceptuel et détermine le type logique de données à extraire.
- Le système examine la description physique de la base en rapport avec la requête logique et détermine le (ou les) enregistrement(s) physique(s) à lire.
- Le système lance une commande au système d'exploitation pour rechercher physiquement l'enregistrement désiré.
- Le système d'exploitation, par le biais de ses méthodes d'accès, accède à l'enregistrement physique.
- Les données demandées sont transférées dans les buffers, ou mémoires tampons.
- Le SGBD, à partir d'une comparaison entre le schéma logique global (conceptuel) et le sous-schéma externe de l'application A, extrait des données stockées dans le buffer, l'enregistrement logique réclamé par le programme d'application. Il effectue également les transformations éventuelles de format.
- Le SGBD transfère les données des buffers dans la zone de liaison du programme d'application A.
- Le SGBD fournit également des informations "d'état" au programme d'application, lui signalant en particulier les erreurs éventuellement constatées au cours du processus d'extraction.
- Le programme d'application, qui dispose des données et d'informations de "service" en assure la bonne exploitation !

- Les ordres d'écriture dans la base physique sont traités par un processus similaire, toute modification ou adjonction étant en général précédée d'une opération de lecture.
- A signaler que, dans la majorité des cas, le SGBD doit traiter simultanément plusieurs demandes de données en provenance de plusieurs programmes d'application, utilisant plusieurs schémas externes différents.

ig Les langages d'un SGBD

- Cette présentation des SGBD fait apparaître la nécessité de bien différencier deux étapes :

- la définition des données par l'administrateur de la base (DBA)
- leur utilisation par les utilisateurs ou les programmeurs d'application.

- Le SGBD met donc à disposition deux types de langage : LDD et LMD

Langage de Description de Données : LDD

- Il permet de décrire précisément la structure de la base et le mode de stockage des données. Alors que l'utilisation de fichiers permet seulement une description de données interne au programme, dans une approche Base de Données, on effectue la description de toutes les données une fois pour toutes : elle constitue l'ensemble des **tables et dictionnaires de la base**, son **schéma** (terminologie CODASYL).

- En particulier, il précise la **structure logique des données** (nom, type, contraintes spécifiques...), la **structure physique** (mode d'implantation sur les supports, mode d'accès), la définition des **sous-schémas ou "vues".**

Langage de Manipulation de Données : LMD

L'utilisation d'une BDD suppose un grand nombre d'utilisateurs, souvent non informaticiens, ayant des tâches et des besoins variés auxquels le LMD doit pouvoir répondre. Le SGBD fournit deux niveaux d'accès :

- le langage d'interrogation, ou langage de requête interactif

évite le recours à des langages généraux de programmation. Il doit avoir une syntaxe souple, si possible graphique, être accessible aux non-spécialistes et permettre la formulation de demandes utilisant des critères variés et combinés.

- le langage hôte

pour les traitements réguliers, le SGBD doit fournir une interface permettant l'utilisation de la base à l'aide des langages procéduraux (COBOL, Pascal, C/C++….), en incorporant les requêtes dans des programmes classiques.

Classification des LMD

- langages navigationnels (ex : SYMBAD)

dans les SGBD hiérarchiques ou réseaux. Les requêtes du langage décrivent les chemins d'accès aux différentes données, celles-ci étant généralement chaînées entre elles.

- langages algébriques (ex : SQL)

dans les SGBD relationnels. Ils utilisent, pour fournir des résultats aux requêtes, les opérateurs de l'algèbre relationnelle.

CHAPITRE ROLE DE L'ADMINISTRATEUR DE LA BASE

- Résumé des fonctions d'un SGBD :

- description de la structure de la base : schéma interne, conceptuel, sous-schémas externes.

- organisation du stockage physique

- manipulation des informations : sélection, extraction, mise à jour…

- protection des données : pour personnaliser les accès à la base, il faut identifier l'utilisateur (code et mot de passe) et vérifier qu'il est autorisé à effectuer les traitements demandés (contrôle des droits d'accès).

- sécurité, restauration : possibilité de reconstituer la base dans un état satisfaisant après tout incident

- optimisation des ressources, tenue d'un journal de tous les événements : le logiciel doit fournir des statistiques précises sur l'état de la base et permettre des réorganisations physiques périodiques qui éviteront la dégradation des performances globales du système.

- intégrité des données : cohérence des informations les unes par rapport aux autres

---

- L'essentiel de la mise en oeuvre de ces fonctions revient à une personne appelée **administrateur** de la BDD qui doit :

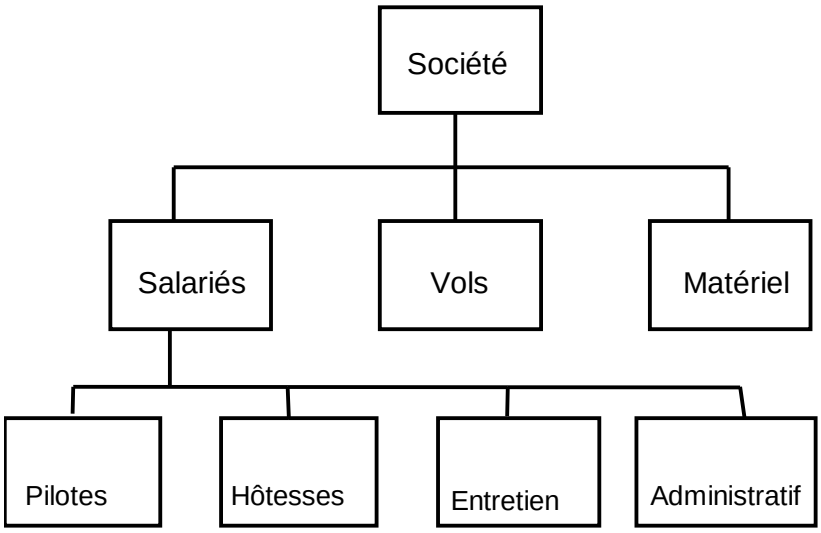
- intervenir en tant que conseil lors de l'étape conceptuelle de l'analyse : responsabilité de gestion des données
- décider des techniques d'accès et de l'implantation physique
- gérer les diverses autorisations d'accès
- définir les stratégies de reprise en cas d'incident
- suivre régulièrement les performances du système et réaliser en conséquence les modifications ou évolutions qui s'imposent.

CHAPITRE PRINCIPAUX MODELES LOGIQUES

Les trois principaux modèles sont, dans l'ordre chronologique de leur arrivée sur le marché, le modèle hiérarchique, le modèle réseau (ou navigationnel), le modèle relationnel.

3g Le modèle hiérarchique

Exemple : le Système d'information d'une compagnie aérienne



- L'ancêtre le plus répandu est le SGBD IMS (Information Management System), développé et commercialisé par IBM dans les années 70

- Caractéristiques générales du modèle :

- Forte dépendance entre la description de la structure des données et la manière dont celles-ci sont enregistrées sur le support physique.
- Les éléments de base du modèle sont des enregistrements logiques reliés entre eux pour constituer un arbre ordonné.
- Les entités (ou **segments**) constituent les noeuds, celui de plus haut niveau portant le nom de racine ; les branches (pointeurs logiques entre entités) constituent les **liens**. Chaque segment est une collection d'objets appelés champs (ou **fields**).
- Chaque segment a obligatoirement un père (sauf la racine), et peut avoir plusieurs fils.

- Avantages :

- rigueur des structures et des chemins d'accès
- simplicité relative de l'implémentation
- adéquation parfaite du modèle à une entreprise à structure arborescente.

- Inconvénients :

- les accès se font uniquement depuis la racine
- la structure interdit les liens N:M, ne permettant que le lien 1:N. La représentation d'autres relations impose de ce fait une redondance de l'information.

Exemple : comment représenter dans ce modèle, un parc de véhicules et un ensemble de chauffeurs, chaque chauffeur pouvant conduire plusieurs véhicules, et un véhi cule pouvant être conduit par plusieurs chauffeurs ?

- les "anomalies" que l'on constate lors des opérations de mise à jour (insertion, destruction, modification) : l'élimination d'un noeud entraîne l'élimination de tous les segments de niveau inférieur qui lui sont rattachés (risque de perdre des données uniques)
- indépendance logique très réduite : la structure du schéma doit refléter les besoins des applications.
- pas d'interface utilisateur simple.

4g Le modèle en réseau

- Evolution du modèle hiérarchique intégrant les résultats du travail du groupe CODASYL (comité de langage de programmation), qui avait démarré l'étude d'une extension de COBOL pour manipuler les bases de données. En 1969, il donne ses premières recommandations concernant syntaxe et sémantique du LDD et du LMD.

- Même si cette vue est un peu simplificatrice, une base en réseau peut être décrite comme un certain nombre de fichiers comportant des références les uns vers les autres. Les entités sont connectées entre elles à l'aide de pointeurs logiques :

- un enregistrement d'un ensemble de données A est associé à une série d'enregistrements (ou records) d'un autre ensemble de données B. On constitue ainsi des SET, ou COSET, structure fondamentale du modèle en réseau
- le lien entre les enregistrements de A et ceux de B est 1:N
- le COSET comporte un type d'enregistrement "propriétaire" (l'enregistrement de A est dit OWNER) et un type d'enregistrement "membre" (les enregistrements de B sont MEMBER).

- Avantages et inconvénients du modèle :

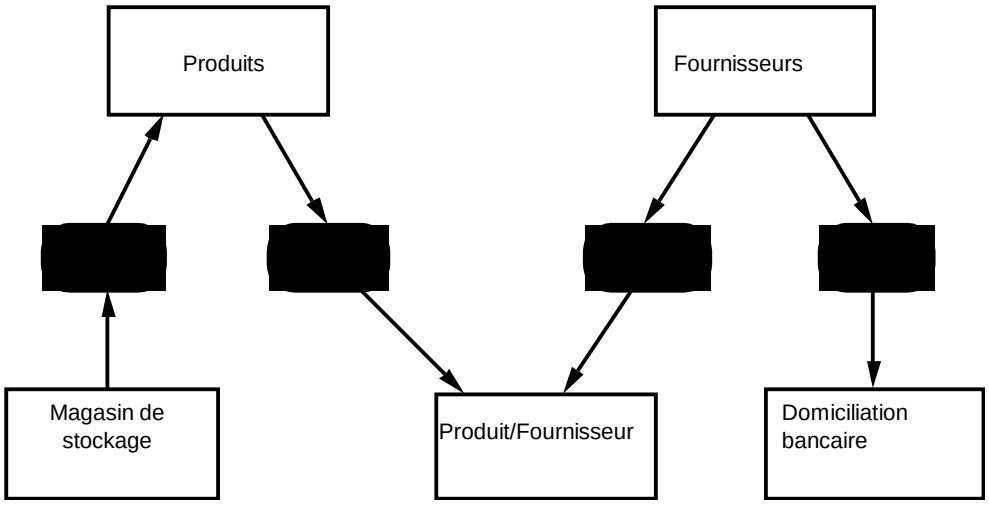
- aucune restriction dans la conception : un type de "record" peut à la fois être propriétaire et membre de plusieurs sets
- représentation naturelle des liens maillés N:M
- pas d'anomalies pour les opérations de stockage
- commercialisation importante des systèmes correspondants (DMS, IDMS, TOTAL, IDS II, SOCRATE...),

MAIS

- pas d'indépendance par rapport aux stratégies d'accès
- procéduralité importante des langages de manipulation ; l'utilisateur doit "naviguer" dans le réseau logique constitué par les enregistrements et les chaînes de pointeurs.

- *Exemple : schéma représentant le sous-système d'information*

produits / magasins de stockages / fournisseurs / domiciliations bancaires



6g Le modèle relationnel

- C'est un article publié en 1969 par un mathématicien du centre de recherche IBM, **Codd**, qui définit les bases de ce modèle relationnel. Codd s'est intéressé au concept d'information et a cherché à le définir sans se préoccuper de la technique informatique, de ses exigences et de ses contraintes. Il a étudié un modèle de représentation des données qui repose sur la notion mathématique de "relation". Dans la pratique, une relation sera représentée par une **table** de valeurs.

Exemple: représentation d'une table du personnel

Matricule	Nom	poste	Salaire	N° dept
350	Durand	Employé	8000	320
780	Dupond	Cadre	15000	870
320	Veillon	PDG	25000	400
490	Martin	Cadre	15000	320

Définitions

- Une relation est un ensemble de tuples (lignes), dont l'ordre est sans importance. Les colonnes de la table sont appelées attributs ou champs. L'ordre des colonnes est défini lors de la création de la table.
- Une clé est un ensemble ordonné d'attributs qui caractérise un tuple. Une clé primaire le caractérise de manière unique, à l'inverse d'une clé secondaire.
- On dit qu'un attribut A est un **déterminant** si sa connaissance détermine celle de l'attribut B (B dépend fonctionnellement de A).

Caractéristiques du modèle

- Schéma de données facile à utiliser : toutes les valeurs sont des champs de tables à deux dimensions.
- Améliore l'indépendance entre les niveaux logique et physique : pas de pointeurs visibles par l'utilisateur.
- Fournit aux utilisateurs des langages de haut niveau pouvant éventuellement être utilisés par des non-informaticiens (SQL, L4G) et un ensemble d'opérateurs basé sur l'algèbre relationnelle : union, intersection, différence, produit cartésien, projection, sélection, jointure, division.
- Optimise les accès aux bases de données
- Améliore l'intégrité et la confidentialité : unicité de clé, contrainte d'intégrité référentielle
- Prend en compte une variété d'applications, en gestion et en industriel
- Fournir une approche méthodologique dans la construction des schémas.

CHAPITRE CONCEPTION DE BASES DE DONNEES

7. Les Formes normales

Les formes normales permettent de construire un schéma conceptuel correct à partir des relations « brutes » issues des données recueillies auprès des clients.

1ère forme normale

Une relation est dite en première forme normale si chaque attribut possède une seule valeur (ce qui exclut les groupes), et si elle admet une clé primaire.

Exemple:

L'exemple porte sur un ensemble de données concernant des tests de types différents, effectués sur les éléments matériel d'un système de production :

R (libellé matériel, code marque, libellé marque, type de test, date du test, résultat du test)
n'est pas en 1ère forme normale car aucun attribut ne peut être clé primaire : le libellé matériel peut être identique pour plusieurs éléments.

R (code matériel, libellé matériel, code marque, libellé marque, code type de test, libellé du test, date du test, résultat du test)
n'est pas en 1ère forme normale car on peut faire plusieurs tests sur un même matériel, ce qui exige de répéter les informations "code type de test", "libellé du test", "date du test", "résultat du test", dans un même nuple.

La relation doit être éclatée en deux, pour être exprimée en 1ère forme normale :

R-MATERIEL (code matériel, libellé matériel, code marque, libellé marque)
R-TEST (code matériel, code type, libellé test, date du test, résultat du test)

Les deux relations ne comportent que des attributs sans répétition. Dans R_TEST, la clé primaire est composée de "code matériel" et "code type" : un type de test peut concerner plusieurs matériels, un matériel peut être testé plusieurs fois, mais chaque matériel ne subit qu'une fois un type de test donné.

2ème forme normale

Une relation est dite en deuxième forme normale si elle est en première forme normale, et si tout attribut n'appartenant pas à la clé primaire ne dépend pas que d'une partie de cette clé.

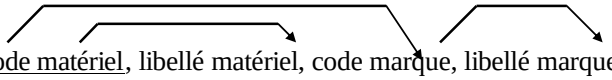
R-TEST(code matériel, code type, libellé test, date du test, résultat du test)
n'est pas en 2ème forme normale car l'attribut "libellé test" ne dépend que du "code type" et pas du "code matériel" ;

La relation doit éclatée en deux, pour être exprimée en deuxième forme normale :

R-TEST (code matériel, code type, date du test, résultat du test)
R-TYPETEST (code type, libellé test)

3ème forme normale

Une relation est dite en troisième forme normale si elle est en deuxième fo rme normale, et si toutes les dépendances fonctionnelles issues de la clé primaire sont directes



R-MATERIEL(code matériel, libellé matériel, code marque, libellé marque)

La dépendance entre "code matériel" et "libellé marque" n'est pas directe, "libellé marque" est en dépendance fonctionnelle directe avec le "code marque".

La relation doit être éclatée en deux, pour être exprimée en troisième forme normale :

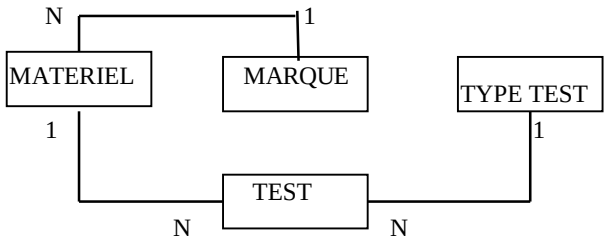
R-MATERIEL(code matériel, libellé matériel, code marque)
R-MARQUE(code marque, libellé marque)

Le schéma conceptuel final de la base de données est donc :

R-MATERIEL (code matériel, libellé matériel, code marque)
R-MARQUE (code marque, libellé marque)
R-TYPETEST (code type, libellé test)
R-TEST (code matériel, code type, date du test, résultat du test)

Commentaires:

- Le schéma conceptuel fait apparaître 3 relations entités : **R-MATERIEL, R-MARQUE, R-TYPETEST**
- et la relation association **R-TEST** qui réalise le lien **Matériel <--> Type test** de type N:M
- Le lien fonctionnel **Matériel <--> Marque** de type N:1 est réalisé par la présence du "code marque" dans la relation R-MATERIEL.



-S Démarche de conception

Concevoir une base de données relationnelle, c'est établir pour le système d'information étudié, les relations entités et les relations associations en troisième forme normale.

1ère étape :

Etablir les schémas externes, c'est-à-dire lister les données nécessaires à chaque utilisateur de la future base.

2ème étape :

Etablir le dictionnaire de données en regroupant les schémas externes, en supprimant les redondances et en ne conservant que les informations élémentaires (non déduites). Ceci revient à lister les attributs de la base.

Dictionnaire de données du système d'informations relatif aux tests sur les matériels de production :

code matériel, libellé matériel

code marque, libellé marque

code type de test, libellé du test, date du test, résultat du test

3ème étape :

Etablir les contraintes d'intégrité fonctionnelle (ou dépendances fonctionnelles) entre attributs.

Attributs	En dépendance fonctionnelle avec

Entité = Clé + Attributs dépendants

code matériel	
libellé matériel	*code matériel*
code marque	
libellé marque	*code marque*
code type de test	
libellé du test	*code type de test*

Association

date du test	*code matériel + type de test*
résultat du test	*code matériel + type de test*

4ème étape :

En déduire les relations "entités" et les relations "associations avec attributs" :

- Les entités sont contituées d'une clé primaire et d'un ou plusieurs attributs qui ne dépendent fonctionnellement que de cette clé
- Les associations sont constituées d'une liste d'au moins deux clés représentant des entités, et d'attributs qui dépendent de ces clés

Entités : *Matériel, Marque, Type de test*

Association avec attributs : Test

5ème étape :

Etablir les relations "associations sans d'attributs" en considérant deux cas :

- Il existe un lien fonctionnel N : 1 entre les entités : la clé primaire de l'entité mère devient clé étrangère dans l'entité fille

Exemple: matériel-marque. L'entité "Matériel" dépend (est fille) de l'entité "Marque" : la clé étrangère "code marque" dans "Matériel" pointe sur la clé primaire "code marque" dans "Marque" .

- Le lien entre les deux entités est de type N:M : il faut créer une nouvelle relation association sans attributs, qui contient seulement les clés primaires des deux relations associées.

6ème étape:

Représenter le schéma de la base

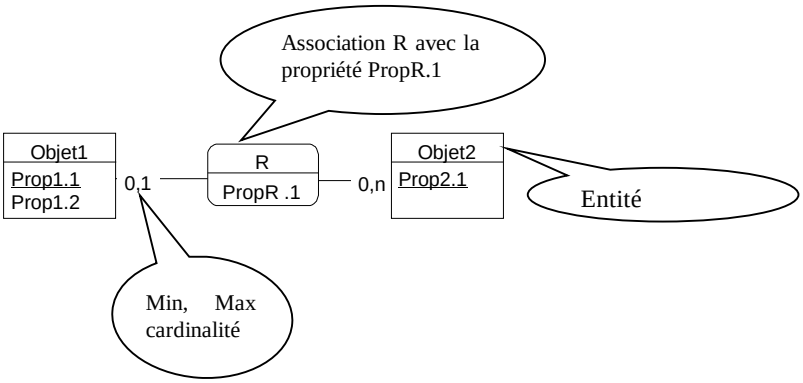
7ème étape:

S'assurer que les relations sont en troisième forme normale.

1g Les phases de la conception avec un symbolisme de type « MERISE »

Présentation

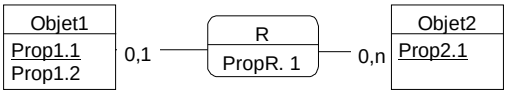
- Sans faire un exposé sur la méthode Merise, ce chapitre voudrait présenter succinctement les différents modèles des données (MCD Conceptuel, MLD Logique, MPD Physique), à titre de comparaison avec la méthode « maison » qui vient d'être exposée.
- Après avoir recueilli les données auprès des clients (étape 1), supprimé les redondances (étapes 2), classé les données selon les dépendances fonctionnelles (étape 3), on construit le modèle conceptuel « entités/associations » (étape 4), où les associations sont des relations valuées, comportant un ou plusieurs attributs :



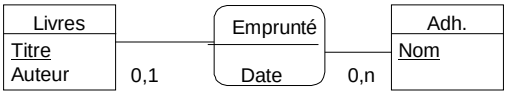
- A partir du modèle conceptuel, on peut déduire le modèle logique et physique par des opérations systématiques (étape 5 et 6) : l'application complète de la méthode Merise garantit l'obtention des formes normales. Le passage du MCD au MLD, puis au MPD dépend de la cardinalité des relations.

Du MCD au MPD, pour un lien fonctionnel (N :1) ou hiérarchique (1 :N)

M.C.D



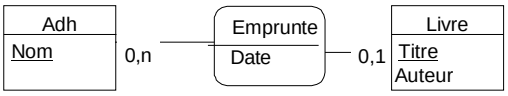
- ***Exemple :***



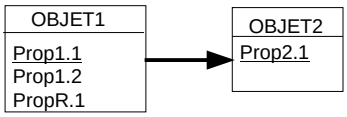
- *L'association* ***« est emprunté le »*** *contient la propriété* ***« Date »***
- ***Cardinalité :*** *Un livre peut être emprunté 0 ou une fois (min = 0, max = 1 dans la notation Merise). Un adhérent peut emprunter de 0 à N livres (min= 0, max= N)*

ð *lien fonctionnel N : 1 dans la notation ANSI-SPARC*

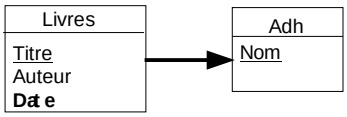
- La relation inverse **« emprunte le »** est une relation hiérarchique 1 :N



M.L.D



Règle **:** les propriétés de l'association glissent du côté 0-1, la flèche pointe vers le côté 0,n



Dans la table Livres, on ajoute la date de l'emprunt, et une flèche vers l'adhérent emprunteur

MPD

Objet1 (Prop1.1, Prop1.2, PropR.1, Prop2.1)

Objet2 (Prop2.1)

Règle : une clé étrangère Prop2.1 pointant sur objet2.Prop2.1 est ajoutée à objet1

Livres *(Titre, Auteur, Date, Nom)*

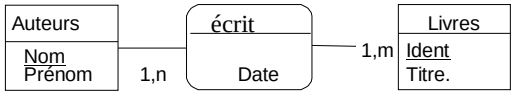
Adherents *(Nom)*

Dans la relation Livres, on ajoute la clé étrangère « Nom », pointant sur « Adherents.Nom »

Du MCD au MPD pour une Relation 0-N : 0-N

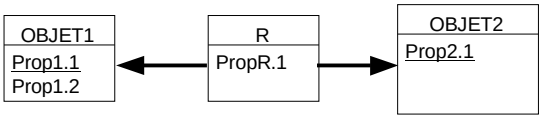
M.C.D

- ***Exemple :***

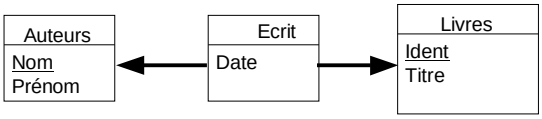


- *L'association* ***"écrit"*** *contient la propriété* ***"Date"*** *(de parution)*
- ***Cardinalité de la relation :*** *Un auteur peut écrire de 1 à n livres. Un livre peut avoir de 1 à m auteurs => lien maillé N : M dans la notation ANSI-SPARC*

M.L.D



Règle **:** l'association devient une nouvelle table et les flèches pointent vers les tables liées



On crée une nouvelle table « écrit », avec la propriété « Date », et des liens vers les clés des entités « Auteurs » et « Livres »

MPD

Objet1 (Prop1.1, Prop1.2)

R1 (Prop1.1, Prop2.1, PropR.1)

Objet2 (Prop2.1)

Règle **:** la relation devient une table dont la clé est la concaténation des clés des deux objets liés.

Exemple :

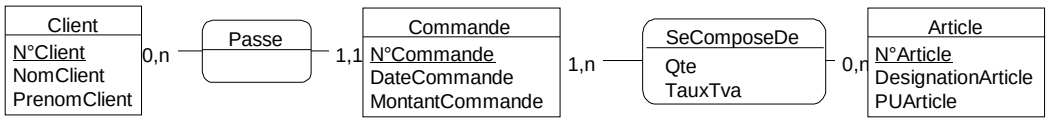
Livres *(Ident, Titre)*

Ecrit (Nom, Ident, Date)

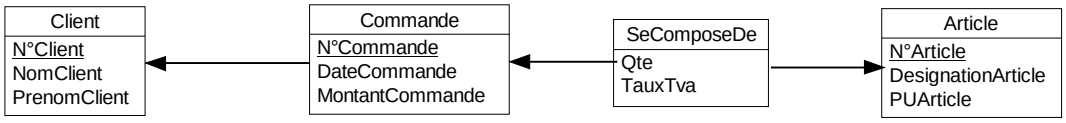
Auteurs *(Nom, Prénom)*

Exemple récapitulatif

MCD



MLD



MPD

Client (N°Client, NomClient, PrenomClient).

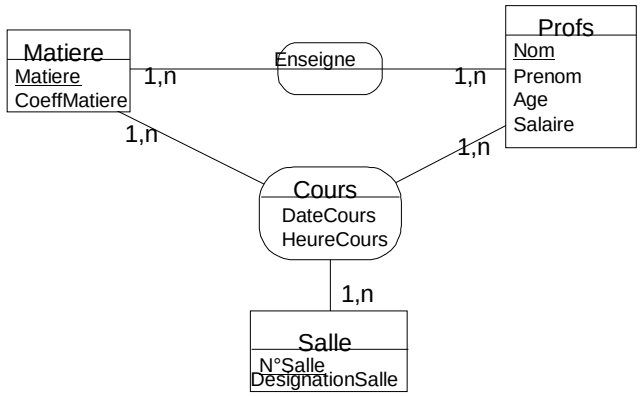
Commande (N°Commande, DateCommande, MontantCommande, N°Client).

SeComposeDe (N°Commande, N°Article, Qte, TauxTva).

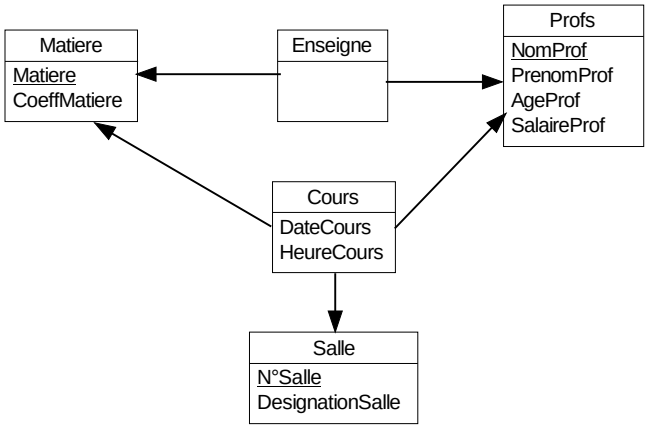
Article (N°Article, DesignationArticle, PUArticle).

Du MCD au MPD, dans une Relation ternaire

MCD



MLD



MPD

Matiere (Matiere, CoeffMatiere).

Enseigne (NomProf, Matiere).

Profs (NomProf, PrenomProf, AgeProf, SalaireProf).

Cours (NomProf, Matiere, N°Salle, DateCours, HeureCours).

Salles (N°Salle, DesignationSalle)

Règle : une relation ternaire devient une table dont la clé est la concaténation des clés des trois objets liés.

CHAPITRE EXERCICES

ENONCE 1

Trouvez les formes normales des relations suivantes (exercices indépendants) :

R1 (N° client, n° produit, quantité commandée, nom produit)

R2 (N° commande, n° produit, quantité commandée)

Dans chaque commande émise, on commande certaines quantités de plusieurs produits. Un même produit peut être commandé dans plusieurs bordereaux de commandes

R3 (N° client, nom client, nom représentant)

Un client peut être suivi par plusieurs représentants de l'entreprise. Pour simplifier, on supposera que les noms des représentant sont uniques, et qu'un représentant ne démarche qu'un seul client.

R4 (N° produit, nom produit, n° atelier, nom chef d'atelier)

Chaque produit est fabriqué dans un seul atelier. Il n'y a bien sûr qu'un chef par atelier !

R5 (N° produit, n° fournisseur, nom fournisseur)

Pour les énoncés qui suivent, il est demandé d'établir le schéma de la base en présentant les relations entités et les relations associations pour chacun des domaines étudié, en suivant la démarche du chapitre 7.2. On représentera le modèle conceptuel avec un symbolisme Merise.

ENONCE 2

- Une société d'édition de livres et manuels universitaires décide de s'informatiser.
- Elle souhaite en particulier automatiser le calcul des droits d'auteur.

- On a relevé lors de l'étude les éléments suivants :

- un auteur est classé dans une seule spécialité
- un livre appartient ou non à une collection. S'il appartient à une collection, il ne peut appartenir qu'à une seule collection.
- les droits d'auteur sont calculés sur le nombre d'exemplaires vendus dans l'année.

- Voici le document envoyé aux auteurs permettant leur paiement :

Relevé des droits d'auteur

Année 1995

Auteur n° 73

Monsieur Dupont Jean
3 rue des alouettes
25 BESANCON
N° SS 1520373265005 Spécialité: Mathématiques

N° ouvrage: 850	Titre: Mathématiques pour la gestion
N° Collection: 6	Collection: Mathématiques appliquées
Nombre d'ouvrages vendus: 800	Taux droits d'auteur: 8%

N° ouvrage: 647	Titre: Mathématiques pour tous
N° Collection: 6	Collection: Mathématiques appliquées
Nombre d'ouvrages vendus: 1000	Taux droits d'auteur: 10%

Total général : 11400 F
Taxe (5%) : 570 F

NET A PAYER 11970 F

ENONCE 3

Une troupe théatrale se produit dans le monde entier. L'équipe établit régulièrement le document décrit plus loin. Elle vous demande de l'analyser en vue d'une informatisation.

Date du jour RECAPITULATIF DES REPRESENTATIONS PAR AUTEURS

AFRIQUE AMERIQUE ASIE EUROPE OCEANIE

nom de l'auteur

titre de la pièce	nbre représ.	date	nbre représ	date	nbre représ	date	nbre représ	date	nbre représ.	date

nom de l'auteur

titre de la pièce	nbre représ.	date	nbre représ.	date	nbre représ.	date	nbre représ.	date	nbre représ.	date

ENONCE 4

Un grand garage se propose d'informatiser son système de gestion en introduisant une base de données pour l'établissement des factures. Voici l'image du document facture:

GARAGE
CHARLES

FACTURE N°: 87234B
DATE FACTURE: 30/08/95

Monsieur DUPONT Jacques
Tel: 89.52.15.32
3 rue des roses

Intervention N° 37843
Date entrée véhicule: 25/08/95
Date sortie véhicule: 26/28/95

Véhicule N° 6837 VH 68
Marque: RENAULT Type R21
Année de mise en service: 1990

CODE OPERATION	LIBELLE	CODE OUVRIER	COUT STANDARD	FOURNITURES ou PIECES
62	Réglage Allumage	G12	52.00	7.00
27	Vidange Graissage	G23	78.00	373.00

TOTAL H.T.	510.00 F
MONTANT TVA	102.00 F
MONTANT TTC	612.00 F

- Pour élaborer ce document, on a relevé les points suivants :

- Une entrée d'un véhicule reçoit un numéro d'intervention ;
- Chaque intervention est formée d'une suite d'opérations ;
- Chaque opération correspond à un travail codifié et affecté d'un coût standard défini par le garage.
- Pour une opération donnée sur un véhicule donné, un seul ouvrier intervient.